

# **FEASIBILITY STUDY FOR A PERSONAL-COMPUTER BASED HEAD-SPINE MODEL**

John B. Bomar, Jr., Ph.D.  
David J. Pancratz, MSME  
Linda J. Rogers, MSCS

Biodynamic Research Corporation  
9901 I.H. 10 West, Suite 1000  
San Antonio, Texas 78230

December 1995

Final Report for period May 1995 to December 1995

Approved for public release; distribution is unlimited.

Prepared for:  
ARMSTRONG LABORATORY  
Human Systems Center  
United States Air Force  
Brooks AFB, Texas 78235-5000

19990329 050

DTIC QUALITY INSPECTED 4

## NOTICES

This final report was submitted by Biodynamic Research Corporation, 9901 IH 10 West, Suite 1000, San Antonio, Texas under contract F41624-95-C-6003 with the Armstrong Laboratory, Human Systems Division, Wright Patterson Air Force Base, Ohio. Dr. Louise Obergefell was the Project Technical Director.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise as in any manner construed, as licensing the holder, or any other person or corporation; or conveying any rights of permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed and it is releasable to the National Technical Information Service (NTIS), where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Please do not request copies of this report from the Human Systems Division. Registered DTIC users may obtain copies from DTIC. Address your request for additional copies to:

Defense Technical Information Center  
Cameron Station  
Alexandria, VA 22304-6145

If your address has changed, if you wish to be removed from our mailing list, or if our organization no longer employs the addressee, please notify AL/XAP, Brooks AFB, TX 78235-5000 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

## TABLE OF CONTENTS

FIGURES .....	vii
ACKNOWLEDGEMENTS .....	viii
SECTION 1 .....	9
1.0 Summary .....	10
SECTION 2 .....	13
2.0 Introduction .....	14
SECTION 3 .....	17
3.0 Technical Background and Literature Review .....	18
SECTION 4 .....	21
4.0 Creating a PC Version of the Head-Spine Model .....	22
4.1 Status of the Model in May, 1995 .....	24
4.2 Porting the HSM Code to and SGI Indigo and PC-DOS Computer .....	26
4.3 Evaluation of HSM Code .....	28
4.4 Head-Spine Model Enhancements .....	30
4.5 Operational HSM Simulations .....	32
SECTION 5 .....	35
5.0 Design of a Nw User Interface for the HSM .....	36
SECTION 6 .....	39
6.0 Head-Spine Model Validity .....	40
SECTION 7 .....	45
7.0 Recommendations .....	46
7.1 Plan and Establish the Simulation Validation Protocol .....	46
7.2 Create a Database of the Geometry and Materials Properties Data .....	49
Contained in the AAMRL HSM	
7.3 Create User Friendly Interface .....	50
7.4 Re-Code the HSM for the PC Environment .....	50
7.5 Validation of the HSM-PC Simulation .....	52
7.6 Conduct Parametric Studies with the HSM-PC .....	52
7.7 Document the HSM-PC and Its Applicability .....	54
SECTION 8 .....	55
8.0 References .....	57

APPENDIX A - HSM Code for a PC-DOS Computer

APPENDIX B - Visual Basic Code for the Input File Interface

APPENDIX C - Inventory of Files

## FIGURES

Figure 1-1.	Summary of Results .....	11
Figure 2-1.	The HSM Final Report .....	15
Figure 3-1.	Literature Search Summary .....	19
Figure 4-1.	Creating a PC-Version of the HSM .....	23
Figure 4-2.	Inventory List .....	25
Figure 4-3.	Process of Porting HSM Code .....	27
Figure 4-4.	Examples of HSM Code .....	29
Figure 4-5.	Examples of Modified Code .....	31
Figure 4-6.	Error Time History .....	32
Figure 5-1.	Opening an Input File .....	37
Figure 5-2.	Input File Data Lines .....	37
Figure 5-3.	An Example of Plotted Output .....	37
Figure 6-1.	HSM Validation Issues .....	42
Figure 7-1.	Technical Objectives .....	47
Figure 7-2.	Simulation Validation Process .....	47
Figure 7-3.	Assessment Team Representatives .....	48
Figure 7-4.	Data Flow of Original HSM .....	51
Figure 7-5.	Overall Program Structure .....	51
Figure 7-6.	Roadmap to Development Process .....	53

## ACKNOWLEDGMENTS

The members of the research team express their deep appreciation to the many individuals and organizations making contributions to this research program. Without the timely assistance given so generously by everyone, the research program and this report could not possibly have been accomplished.

The research team wishes to express their gratitude for the cooperation, support, and guidance of the Air Force Technical Project Manager Dr. Louise Obergefell of the Armstrong Laboratory.

The assistance provided by Mr. Bob Gallaway and Ms. Annette Rizer both of SRL, Incorporated was very helpful. Ms. Rizer's help in getting the archived HSM materials organized and shipped to BRC was essential in making early progress on the program. Mr. Gallaway's suggestions relating to user interface improvements were incorporated in the recommendations for future improvements to the HSM.

Finally, the research team gives a very special thank you to the support staff at BRC. Without the efforts and dedication of Celina Canales and Anne Hofmeister in operations support, this research program could not have been reported effectively. Appreciation is also due to Dr. James Raddin and Eric Weiss for technical support, John Martini for illustration support, and Patricia Perret for information services support. And last, but not least a special thanks is given to Sue for her support during this research program. A special thanks is given to Sue for her support during this research program.

## **SECTION 1**

### **SUMMARY**

## 1.0 Summary.

Biodynamic Research Corporation (BRC) of San Antonio, TX, completed an SBIR Phase I project to port the Air Force's Head-Spine Model (HSM) to a PC-DOS environment and provide a recommended roadmap for the future of the HSM. The impetus for this project was the Air Force's desire to have a software tool capable of modeling the internal forces and motions of the human head and spine during impulsive acceleration events, particularly aircraft ejections. Although models exist to predict the gross motion of a human under acceleration loading such as the Air Force Articulated Total Body (ATB) model, Dynaman, and MADYMO, the Head-Spine Model (HSM) is the only tool able to provide estimates of internal forces. For this reason, the HSM could be valuable to the Air Force and other scientists for simulating acceleration environments.

The HSM was presented to BRC as archived FORTRAN files that had been unused for nearly 8 years; none of the scientists originally involved in the development of the HSM were available for consultation. The program, originally designed to run on a Concurrent mainframe computer, had been ported to a Silicon Graphics Incorporated (SGI) Unix workstation. BRC was able to successfully compile and operate the model on an SGI workstation and port the code to an MS-DOS compatible PC computer. The DOS and Unix versions of the model produced essentially the same output for three different simulations.

Although the code transfer was successful, BRC discovered several "problems" with the Head-Spine Model. For one, the code was written in such a way that it required both an input file of material and geometry data for each simulation and a FORTRAN source file or set of source files that had to be recompiled for each simulation. This handicap made it difficult to change the characteristics of a simulation using the HSM. The model was also coded so that it was difficult to understand, debug, and modify. There are several areas in the model code that BRC suspects are incorrectly implemented, however, there is no way to confirm this because of the lack of comments and documentation. An attempt was made to restructure some of the FORTRAN model code in modern structured Fortran 77 code.

To assist in understanding the input and output data for an HSM simulation, BRC developed a custom software program that reads the input file and displays it for modification, runs the HSM model, and reads the output data and allows it to be graphed and printed. Unfortunately, as noted earlier, the HSM still requires recompilation.

It is BRC's belief that the material and geometry data of the HSM model, as well as some of the model algorithms and logic, can best be used by recreating the model in an object-oriented software language such as C++ or Fortran 90. The cost and effort required to understand the current version, debug coding and algorithm errors, and document the code is far greater than simply extracting the useful data and starting over. A roadmap for developing the PC-based HSM is presented at the end of this report. Figure 1-1 summarizes the Phase I Results.



**Figure 1-1 Summary of Results**

Porting the HSM to a PC
<ul style="list-style-type: none"><li>• Compiled and is executable on a PC and Silicon Graphics.</li><li>• Nearly identical results on a PC and SGI.</li><li>• Three simulations conducted.</li></ul>
HSM Improvements
<ul style="list-style-type: none"><li>• Graphical user interface created for changing input file, running the model, and displaying outputs.</li><li>• Structured Fortran techniques incorporated into code.</li><li>• Apparent errors found in the original code.</li></ul>
Critique of the Model
<ul style="list-style-type: none"><li>• Difficult to validate because of the number of degrees of freedom and many parameters.</li><li>• Is the only model that provides internal force and motion computations.</li></ul>
Recommendations
<ul style="list-style-type: none"><li>• The Head-Spine Model be re-coded with a modern, object-oriented approach.</li><li>• Use existing material and geometry data from the original HSM.</li><li>• Validate the model for certain input forces and accelerations.</li><li>• Commercialize the use of the model.</li></ul>



## **SECTION 2**

### **INTRODUCTION**

## 2.0 Introduction.

This final report concludes an effort by Biodynamic Research Corporation (BRC) of San Antonio, TX, to port the Air Force Head-Spine Model to a PC-DOS computer and conduct a feasibility study for making the HSM a useful desktop tool. The effort was conducted under Contract F41624-95-C-6003 through the Armstrong Laboratory and is entitled "Feasibility Study for a Personal-Computer Based Head-Spine Model."

This effort emanates from the Air Force's desire to develop a mathematical model of the internal forces and motions that occur within the human head and spine during impulsive acceleration events and aircraft ejections in particular. Existing tools such as the Articulated Total Body (ATB) model, Dynaman, and MADYMO are able to simulate a human's gross motion to forces and accelerations, but are either unable or invalidated for computing the internal dynamic and kinematic quantities. These internal forces and stresses can be used as predictors for injury. A validated HSM would be a useful tool for Air Force and commercial researchers and scientists in a variety of fields.

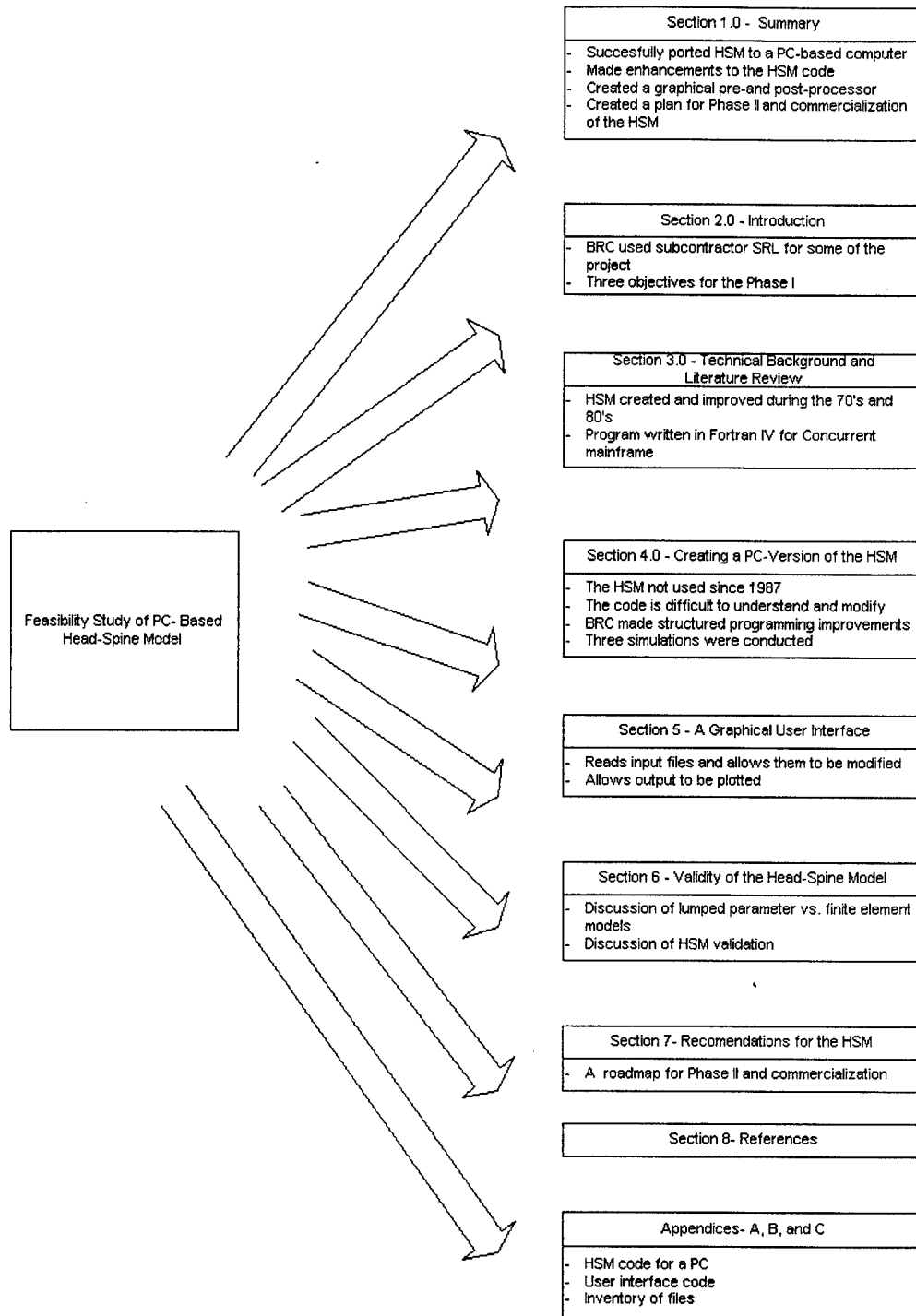
The specific objectives of the Phase I effort are listed below:

- To port the HSM from a Silicon Graphics Unix computer to a desktop PC-compatible computer.
- To conduct a feasibility analysis of improving the software implementation of the model, adding a user-interface, and documenting the model.
- To evaluate the merits and limitations of the model, suggest ways to enhance the realism of the model or decrease its complexity, and identify a protocol for validating the model operation under certain conditions.

BRC performed a majority of the work for this contract. Systems Research Laboratory (SRL) of San Antonio, TX, assisted with some of the analysis to identify an ideal user interface for the model.

This final report documents and summarizes the work generally according to the objectives outlined above. The layout of the report is summarized in Figure 2-1.

**Figure 2-1 The HSM Final Report**





## **SECTION 3**

### **TECHNICAL BACKGROUND AND LITERATURE REVIEW**

### 3.0 Technical Background and Literature Review.

The Air Force Head-Spine Model (HSM) was initially developed for the purpose of modeling pilot ejections from aircraft. In 1973-5, a three-dimensional model of the human spine and head was created at the University of Illinois, sponsored by the Air Force Aerospace Medical Research Laboratory.<sup>1</sup> The model consists of rigid bodies, which represent skeletal segments, and deformable elements, which represent ligaments, cartilaginous joints, viscera, and connective tissues. The model can be separated into sub-models to study the response of different body segments to accelerations and loads. The original HSM is theoretically sufficiently general to be applicable to other acceleration environments. A technique for modeling other aspects of the environment, such as seat geometry and restraint harness, are also considered in the HSM. Output of the model includes the forces and moments acting on the elements and the kinematics of the model elements.

The HSM was refined in 1976-7 during a second Air Force project.<sup>2</sup> During this project, four different versions of the model were created ranging from 32 to 252 degrees of freedom. The researchers also attempted to refine and validate the model using two methods -- by comparing the model frequency response to the experimentally determined frequency response of humans to vertical excitation and by creating head-spine models for other primates and comparing the model output to experimental data. The work with the primate model also provided a methodology for scaling dynamic response and injury data between the primates and humans. Finally, a spinal injury criteria were developed that use the computed stress in vertebral bodies to predict the likelihood of vertebral body failure.

A database of head and cervical spine geometry and stiffness data was collected in a third project from 1978-80.<sup>3</sup> The geometry of the vertebrae and points of attachment of muscles, discs, and ligaments were obtained. Stiffness data were obtained or estimated from the literature. Inertia values were developed by approximating the geometry of the vertebrae and applying a uniform density. Simulations with the model were in relatively good agreement with experimental data for the first 150 milliseconds of response. After this time, the model output showed significant discrepancies which the authors attributed to poor modeling of the major muscles. Although this project focused on the head and cervical spine, it used portions of code from the entire HSM model.

One final revision to the Head-Spine Model was an effort from 1980-4.<sup>4</sup> During this project, a model of the diaphragm was incorporated into the HSM that better represented the vertical load path through the viscera, abdomen, and rib-cage region. Additional work was done to create a proposed injury criterion for the cervical spine. Simulations with the head and cervical spine model in the fore-aft and side-to-side directions showed a good agreement to experimental data, which led the authors to believe the model was a viable tool for prediction of head-cervical spine kinematics in three dimensions.

Figure 3-1 summarizes our literature search efforts and results.



**Figure 3-1 Literature Search Summary**

Literature Search Topics	
<p>Subject Areas:</p> <ul style="list-style-type: none"> <li>• Head/neck/spine modeling.</li> <li>• Injury modeling and criteria.</li> <li>• Dynamic response of the head/neck/spine.</li> <li>• Air Force Head-Spine Model.</li> </ul> <p>Authors:</p> <ul style="list-style-type: none"> <li>• Abe Privitzer</li> <li>• Jeff Settecerci</li> <li>• T. Belytschko</li> </ul>	
Briefs for Select Related Literature	
1.	"A Model for Analytic Investigation of Three-Dimensional Head-Spine Dynamics," [Ref 1]. Development of the original Head-Spine Model and techniques for modeling environment, restraints, and seats.
2.	"Refinement and Validation of a Three-Dimensional Head-Spine Model," [Ref 2]. Refinement and partial validation of the Head-Spine Model.
3.	"A Dynamic Model of the Cervical Spine and Head," [Ref 3]. Discussion of geometry and material data for the HSM and simulations with the cervical spine and head portion of the model.
4.	"Head-Spine Structure Modeling: Enhancements to Secondary Loading Path Model and Validation of Head-Cervical Spine Model," [Ref 4] Incorporation of abdominal load path and spinal injury criteria into the model.
5.	"A Biomechanical Model of the Human Spinal System," [Ref 6]. Describes a finite-element model of the spinal column, ligaments, muscles, rib-cage, abdomen, and part of the pelvis.
6.	"Tolerance of the Human Cervical Spine to High Acceleration: A Modeling Approach," [Ref 7]. Development of and simulations with a sagittal plane model of the cervical spine.
7.	"An Analytical Model of Intervertebral Disc Mechanics," [Ref 8]. A model of the intervertebral disc which predicts loads within the annulus.
8.	"Cervical Spine Analysis for Ejection Injury Prediction," [Ref 9]. Sagittal plane model for the cervical spine and skull that includes muscles, ligaments, and intervertebral joints.



## **SECTION 4**

### **PORTING THE CODE TO A PC**

#### **4.0 Creating a PC Version of the Head-Spine Model.**

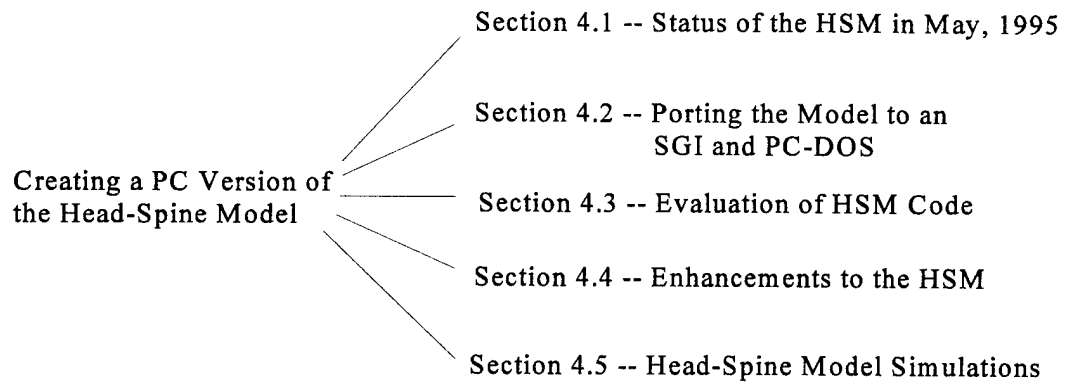
The Head-Spine Model was recompiled on a PC-DOS computer and modified so that it could perform three simulations with identical outputs to the original Unix version of the model. When first presented to BRC, the HSM model had been virtually unused since 1987. The materials that BRC obtained and the status of the HSM model before this project are described in detail in Section 4.1.

BRC had little difficulty in porting the HSM to both a PC-DOS computer and an SGI Indigo computer running Unix. The various minor problems encountered are detailed in Section 4.2. An audit was conducted of the output produced by the new executables and the archived output; the PC-DOS version of the model was found, with minor differences, to compute the same results as the archived executables.

Although porting the code proved fairly easy, the HSM code itself was difficult to understand, debug, and modify. Section 4.3 discusses some of the problems encountered, which were a combination of FORTRAN IV constraints and poor programming techniques. Our efforts to improve the coding of the model are described in Section 4.4. Several elements of structured programming were incorporated into the code, including DO...END DO statements, BLOCK IF structures, and better formatting.

Finally, Section 4.5 describes the simulations that BRC was able to conduct with the model. We were able to successfully execute only three simulations, two of which were very simple. It was discovered that HSM simulations required both a special input file of material and model geometry data, as well as a unique subroutine or set of subroutines that had to be compiled with the remainder of the model. Although other input files were found in the archived code, BRC was unable to make these simulations execute properly. Figure 4-1 presents an overview of the contents of Section 4.

**Figure 4-1 Creating a PC Version of the HSM**



#### **4.1 Status of the Model in May, 1995.**

Before the start of this Phase I SBIR, the Head-Spine Model software and documentation had been virtually unused since 1987. In 1987, a document titled "Directions/Overview for Running the Head Spine Model" was created at WPAFB that summarized the status of the model. This document, along with the user's manual for the model and various technical reports, provided a foundation for BRC to trace the model history and become familiar with the model.

BRC was presented with the documentation and archived code in June of 1995. All of the materials received from the Air Force have been recorded and an inventory of files is attached in Appendix C and a summary of other materials is shown in Figure 4-2. The materials include program listings, technical reports, sample runs, and magnetic media. The files on the magnetic media included the most recent version (1987) of the HSM software for use on a Silicon Graphics workstation and two archived directories of material.

BRC conducted a line-by-line comparison of the magnetic media files and the printed code that was reported to be the "most current version of the HSM" to ensure that the proper version of the software was analyzed. A few minor discrepancies were discovered; however, the code audit confirmed that BRC indeed had the source code corresponding to the "most current version of the HSM."

The code provided to BRC included an executable of the model for SGI Unix. It was possible to execute this file without modification for the "hyb2.inp" input file, which apparently corresponded to a Hybrid II Anthropomorphic Mannikin Head-Neck pendulum test. Results of this simulation are discussed in Section 4.4.

**Figure 4-2 Inventory List**

Inventory Items
<ul style="list-style-type: none"> <li>• Program listing 2/17/81</li> <li>• Plotting program source 11/9/77</li> <li>• Sample run 8/6/87 head &amp; cervical spine, ½ spine input accel. Profile</li> <li>• Sample run 8/19/87 isolated ligamentous spine model 14 G, 100 ms</li> <li>• Sample run 8/6/87 simplified based on SSM deformations</li> <li>• Sample run 9/19/83 Sikorsky IV 003 w/ injury criteria</li> <li>• Sample run 12/7/81 baboon model 1B</li> <li>• Sample run 11/18/80 bird strike simulation</li> <li>• Sample run GHSM rectangular arc profile simulation</li> <li>• Energy absorbing seat design</li> <li>• Sample run 5/12/83 axial load uniform pressure distribution</li> <li>• Fastplot listing and instructions by M. Hoffman</li> <li>• "Head Spine Model Northwestern Version 1978 Input File Description"</li> <li>• "Head Spine Model Input File Description 1979"</li> <li>• "Dynamic Distribution of Stress and Injury Likelihood in the Spine", by M. Hoffman, 1983 (2 copies)</li> <li>• Brown binder -- compilation of materials</li> <li>• HSM user manual by L. Schwer and G. Belytschko, modified by M. Hoffman 4/11/77 (2 copies)</li> <li>• Color coding for HSM, user guide for output, by M. Hoffman 4/11/77 (2 copies)</li> <li>• Tech paper AF binder, "Head Spine Structure Modeling: July 1985 Enhancement to Secondary ...", by T. Belyschko, M. Rencis, and J. Williams</li> <li>• HSM user guide May 1985 by SRL</li> <li>• Sample run 3/12/82, SAP 4 head injury model, 1500 N vertical load</li> <li>• HSM source code listing 10/15/84</li> <li>• Fastplot source code file and variations 11/83</li> <li>• HSM source code listing 8/24/84 and miscellaneous plots</li> <li>• HSM source code for subroutines: Freefds, Injcri, Spinif, Sliders, Baboon, Splmfit</li> <li>• Sample run CR24B 1984 Pre-Crest simulations</li> <li>• Magnetic tape of most current HSM files and two archived directories of older HSM files.</li> </ul>

## **4.2 Porting the HSM Code to an SGI Indigo and PC-DOS Computer.**

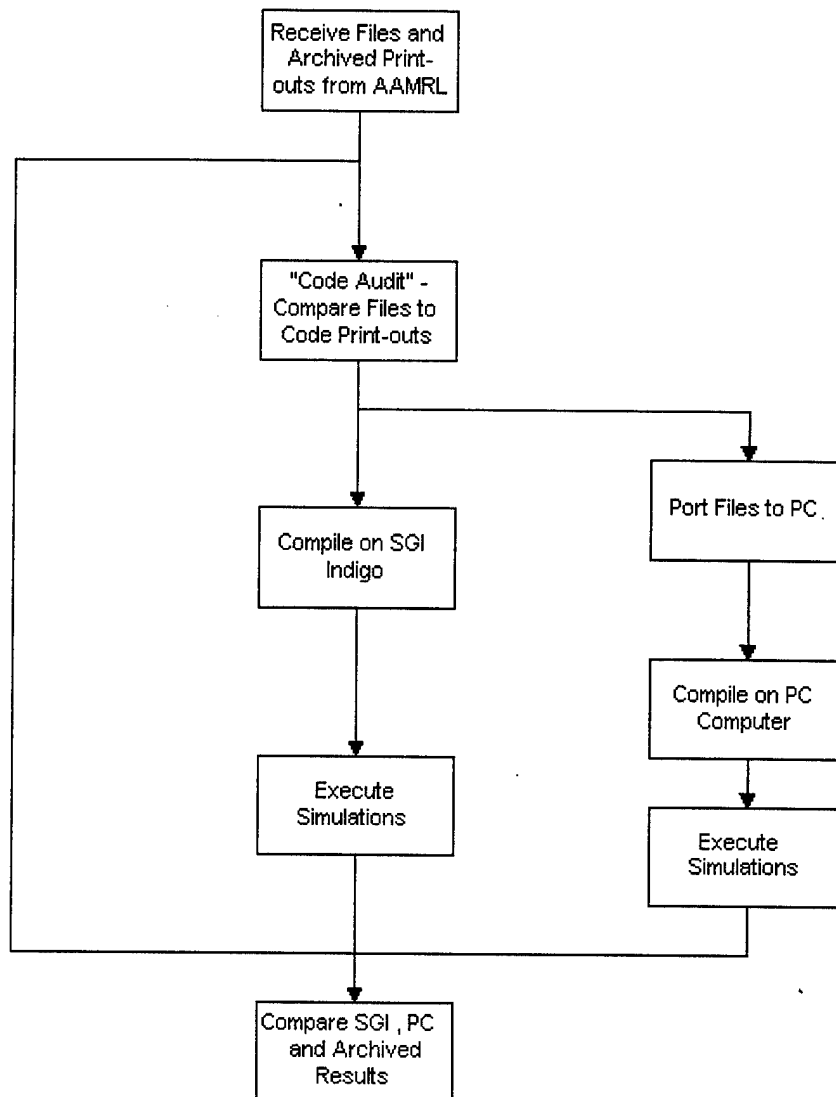
An executable of the HSM code was created for an SGI Indigo workstation and a PC-DOS computer. A similar procedure was followed on both platforms to create the executables. First, subroutines that performed no function or were involved in printing or plotting to an actual device were removed. This eliminated many "unsatisfied external references" errors issued by the compilers. Subroutines were then added and debugged until there were no unsatisfied externals. Most "bugs" were comprised of illegal characters and minor syntax errors which were easily corrected. The final compilation was successful, but produced "run-time errors" which were systematically investigated and eliminated until a program was produced which would execute with the input file "hyb2.inp." The output file from the PC was compared to an existing output file for the "hyb2.inp" which ran on the Silicon Graphics computer via an executable which was provided in the software archives.

The PC version of the model was compiled on an IBM compatible PC running Microsoft Windows 3.1 using Microsoft FORTRAN Powerstation Professional Development System v 1.0. The SGI version of the model was compiled on a Silicon Graphics XS24 Indigo running IRIX Unix v 5.3 and a FORTRAN 77 v 4.0.2 compiler.

The process of porting the HSM code to a PC and verifying proper execution is summarized in Figure 4-3. The modifications made to the code at this point of the project were only those necessary to make the model execute; a discussion of the general condition of the code is presented in Section 4.3. The final code for the PC-version of the Fortran code is listed in Appendix A.



**Figure 4-3 Process of Porting HSM Code**



### 4.3 Evaluation of HSM Code.

The HSM code was originally written in FORTRAN IV and is, in general, difficult to understand, debug, and maintain. When presented to BRC, the HSM code had many problems, some typical of FORTRAN code, such as:

- The use of GOTO statements, which obfuscate the flow of the program. There were many logical loops and branch statements that relied on GOTO's to redirect program flow.
- A lack of substantive comments. There were virtually no useful, descriptive comments in the code.
- A confusing use of COMMON blocks, which sometimes used different variable names in different subroutines.
- Overwriting of memory allocated for an array. While this technique can be used to minimize the use of memory, it appears that data was being overwritten outside of the array bounds, which may have caused erroneous results.
- Numerous calls to blank "do-nothing" subroutines, or subroutines that involved plotting results on antiquated devices.
- Subroutines that contained a significant amount of code, but were not called because of program redirection before the subroutine calls.
- Subroutines that were passed variables that were never used.
- A lack of indenting and spacing in the code, which would have allowed a more intuitive understanding of program flow.

Some of the HSM problems were simply due to the nature of early FORTRAN. For example, constraints in formatting and a lack of structured programming statements resulted in some of the problems above. The lack of comments, questionable use of array memory, and substantial numbers of extraneous variables and subroutines were indicative of the programming style. Some examples of poor coding techniques are displayed in Figure 4-4.

The nature of the HSM code made it very difficult to logically trace program flow. An effort was made to update the code, both to improve the readability and better understand the logic flow. These efforts are described in Section 4.4.

Figure 4-4 Examples of HSM Code

Example of Lack of Formatting

```
STRS(IEE)=.5D0*UKE
IF (NSLIDP.EQ.0) GO TO 6
CALL SLIDER (NSLIDP,FORCD,FINT,SMASS,A1,X1,TIME,BETA,NCP,NASN,INME
1SH,DICOSP,ALPHA,UP,UP2,NPNO,SEATK,UPOLD,SEATWK,SEATEX,V,UPI,VD
2AMP,DELT)
6CALL UPDATE ( NPRI,NDGREE,DELT,X1,XO,V,A1,EULCO(1,1,1),BETA )
CALL FRCIN (NUMEL,NUMNP,NDGREE,XC,YC,ZC,IX,E,X1,FINT,STRS,STRAIN,S
1TRESS,IPT,INDEX,EULCO,SMASS,DICOS,NPRI,AL,JEIGEN,YIPT,ZIPT,XLEN,TH
2CK,INMESH)
WRITE (6,902)
```

Example of Confusing Logic

```
19 I=I+1
IF (I.GT.NEQ) GO TO 21
IF (SMASS(I).EQ.0.) GO TO 20
X1(I)=XO(I)+V(I)*DELT+C1*AO(I)
GO TO 19
20 I=I+5
GO TO 19
21 CONTINUE
```

No Comments

```
DO 90 I=1,NB
FP=DABS(PE(I))/PY(I)
FMZ=DABS(BMZE(I))/BMZY(I)
FTEMP=FP+FMZ
IF(FTEMP.LT.F1(I)) GO TO 81
F1(I)=FTEMP
FA(I)=F1(I)
PF1(I)=PE(I)
BMF1(I)=BMZE(I)
TF1(I)=TYME
81 FMY=0.D0
IF(ISYM.EQ.0) GO TO 90
FMY=DABS(BMYE(I))/BMYI(I)
FTEMP=FP+FMY
IF(FTEMP.LT.F2(I)) GO TO 82
F2(I)=FTEMP
PF2(I)=PE(I)
BMF2(I)=BMYE(I)
TF2(I)=TYME
82 FTEMP=FP+DMAX1(FMY,FMZ)
IF(FTEMP.LT.F(I)) GO TO 90
F(I)=FTEMP
FA(I)=F(I)
PF(I)=PE(I)
BMF(I)=BMYE(I)
IF(FMZ.GE.FMY) BMF(I)=BMZE(I)
TF(I)=TYME
90 CONTINUE
```

#### 4.4 Head-Spine Model Enhancements.

BRC improved the structure of the model FORTRAN code and created an intuitive user interface for modifying the HSM input files. Some aspects of structured FORTRAN were incorporated into the Head-Spine Model code to help better understand the flow of the program.

As was noted in Section 4.3, the HSM code was presented in a way that made it difficult to understand and modify. BRC spent a significant amount of time improving the HSM code so as to better understand the flow and enhance the readability for the future. One of the first improvements was to eliminate calls to blank subroutines and subroutines involved in plotting output. Presumably, the blank subroutines represented areas of expansion for the model that were never undertaken. Numerous variables that were passed into subroutines but not used were also removed. BRC took advantage of compiler INCLUDE statements to replace many of the COMMON blocks which, because of their length, distracted the reader from the function of the subroutines.

BRC also implemented some features of structured programming. Branch IF statements that relied on GOTO's for redirection were changed to BLOCK IF statements. In some cases this proved impossible without rewriting the entire subroutine. DO...END DO statements were used to replace DO...CONTINUE statements. More liberal use of indenting and spacing of code was included to make the program more readable. The use of line numbers was eliminated whenever possible.

No attempt was made to correct several arrays that were suspected of being accessed improperly. Making these changes would have prevented us from understanding the effect of other modifications on the code, since the output of the program would then be different. Several calls to subroutines that were never used because of logic preceding the call to the subroutine were left in place. It is suspected that these subroutines might be required for other simulations, although none of the simulations BRC made operational required them.

Examples of typical HSM code segments before and after the Phase I project are shown in Figure 4-5.

Figure 4-5 Examples of Modified Code

Old Segment of Code

```
19 I=I+1
   IF (I.GT.NEQ) GO TO 21
   IF (SMASS(I).EQ.0.) GO TO 20
   X1(I)=XO(I)+V(I)*DELT+C1*AO(I)
   GO TO 19
20 I=I+5
   GO TO 19
21 CONTINUE
```

New Segment of Code

```
19 I=I+1
   IF (I.LE.NEQ) THEN
     IF (SMASS(I).NE.0.) THEN
       X1(I)=XO(I)+V(I)*DELT+C1*AO(I)
       GO TO 19
     END IF
     I=I+5
     GO TO 19
   END IF
```

Old Segment of Code

```
DO 9 JE=1,NUMEL
  ISEC=IX(11,JE)
  IF(NOPT.EQ.6) IADD=5
  IF (NOPT.NE.2) GO TO 5
  I1=1
  IF (ISEC.NE.0) I1=IPT(ISEC)
  IADD=41+6*I1
  IF (KONTRL(5).GT.0) IADD=41+8*I1
5  INDEX(JE+1)=INDEX(JE)+IADD
  IND=INDEX(JE)-1
  IF (NOPT.NE.2) GO TO 7
  IF (ISEC.EQ.0) GO TO 7
  IF (KONTRL(5).GT.0) KK=KK-I2
  DO 6 M=1,I2
    STRS(KK)=E(3,MTYP)
6  KK=KK-1
7  CONTINUE
```

New Segment of Code

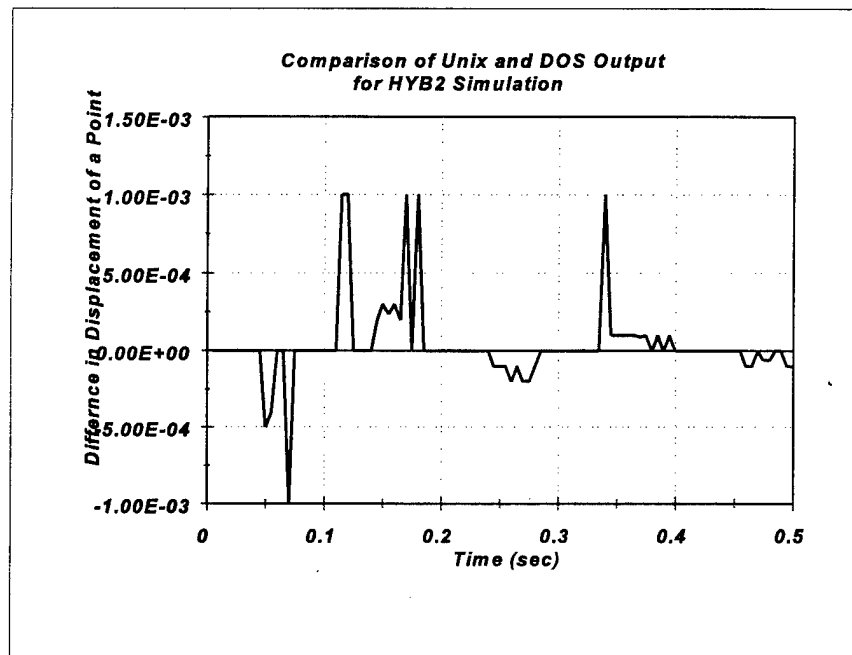
```
DO JE=1,NELE
  ISEC=IX(11,JE)
  IF (NOPT.EQ.2) THEN
    I1=1
    IF (ISEC.NE.0) I1=IPT(ISEC)
    IADD=41+6*I1
    IF (KONTRL(5).GT.0) IADD=41+8*I1
  END IF
  INDEX(JE+1)=INDEX(JE)+IADD
  IND=INDEX(JE)-1
  IF (NOPT.EQ.2 .OR. ISEC.NE.0) THEN
    IF (KONTRL(5).GT.0) KK=KK-I2
    DO M=1,I2
      STRS(KK)=E(3,MTYP)
      KK=KK-1
    END DO
  END IF
```

#### 4.5 Operational HSM Simulations.

As described above, BRC was successful in porting the full HSM model to the PC environment. Unfortunately, as explained below, the HSM is not general. Only three simulations were successfully compiled and executed: two Hybrid II Anthropomorphic Test Device (ATD) Head-Neck simulations and one Crest Ejection Seat simulation. The "Main" program for the version of the HSM which was successfully executed was "WHAM3.FOR." WHAM3.FOR, was compiled on an IBM Compatible PC (Gateway 2000 486-DX2) under Microsoft FORTRAN Powerstation Professional Development System v1.0. Software and run-time bugs were eliminated until we produced a program which would execute with the input file "hyb2.inp".

The output file produced by the PC executable was compared to an existing output file for the "hyb2.inp" which ran on the Silicon Graphics Incorporated (SGI) computer via an executable which was provided in the software archives. There were only very small differences in the numerical values output by each computer. The largest errors found were on the order of  $10^{-3}$ . Figure 4-6 is a plot of the difference in the output position of a point on the spine computed on a PC and that of the same point computed on an execution on the SGI versus time. There was no way to tell which of the two solutions were most accurate, but it was assumed that the SGI was the more accurate because of its increased precision.

**Figure 4-6 Error Time History**



Most other outputs were either matched exactly or were off by a similar differences in values. Similar results were obtained from the input file "hyb22.inp"

The CREST ejection seat simulation was compared to archived outputs and confirmed that the executable was operating properly. The same input files were also executed on a Gateway 2000 P5-90 Pentium PC and results were in agreement with the archived and SGI outputs. Surprisingly, the CREST simulation took only 17 minutes to execute on a Pentium and 42 minutes on the SGI XS24. The run time for a Gateway 2000 DX2-66V (80486 DX2-66MHz) was 1 hour 30 minutes.

During the process of trying to duplicate archived simulations it was discovered that WHAM3.FOR had to be recompiled with different subroutine modules when the object being simulated was changed. This problem has been traced to a subroutine FREED.FOR which is unique to the problem at hand (it apparently sets initial conditions on the model). Thus, it appears that a different, custom FREED.FOR subroutine must be created for each problem type and compiled with the remainder of the source code to build an executable which will properly compute the desired model solution. This is not a good programming technique. The program could be made more general by rewriting the appropriate program sections. Although we examined both the code and the available documentation, BRC was unable to completely discover how the FREED.FOR subroutine is employed within the larger program to set initial conditions nor was it clear how a custom FREED.FOR subprogram is designed.





## **SECTION 5**

### **A NEW METHOD FOR CREATING INPUT FILES**

## 5.0 Design of a New User Interface for the HSM.

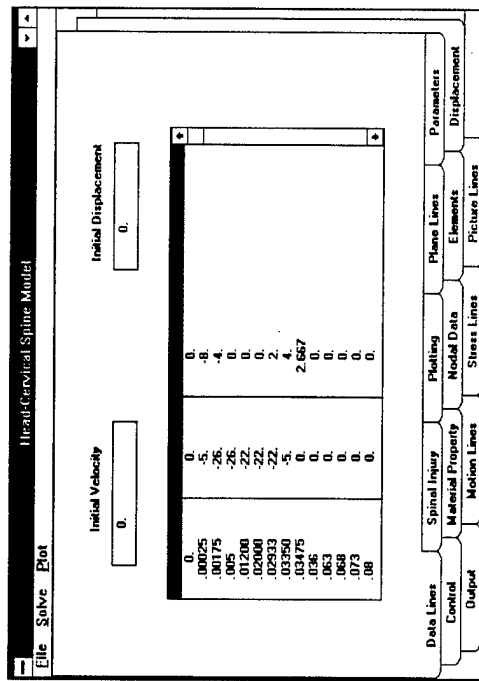
A prototypical intuitive graphical interface was developed by BRC to allow the modification of HSM simulation input files and the display of HSM simulation output. The pre-processor/post-processor module is written in Visual Basic for Windows Version 3.0 and uses three third party custom controls: IndexTab 5.0 for tabbed screens, TrueGrid Pro 2.1 for data editing, and Chart FX 3.0 for the plotting capabilities. With this interface module the user has options to view, edit, and save input files, execute the HSM program, and view plots based on the output of the HSM program. The installation instructions for the User Interface Module and a listing of the source code can be found in Appendix B.

The menu for the User Interface has three options: **File**, **Solve**, and **Plot**. The **File Menu** provides the user with the options to open and save input files. Additionally, the file menu can be used to exit the program. To open an input file, select the **Open** option from the **File** menu as shown in Figure 5-1. The **Open** option brings up a dialog box that assists the user in finding and loading an input file. Once an input file is opened, the user can use the tabs to view the data. Some of the data lines have single records in the input file and others have multiple lines. The tabbed screens display field edit boxes for single record data lines and grids for multiple record data lines. Figure 5-1, which shows the Parameter Data Line, demonstrates a screen with edit boxes. A screen with a grid is presented in Figure 5-2 which displays the screen for the ICIF Data Lines. Subroutine ICIF apparently takes care of converting the input forcing function to the integration routine time scale and integrating accelerations and velocities as necessary to obtain a displacement versus time forcing function for input to the model.

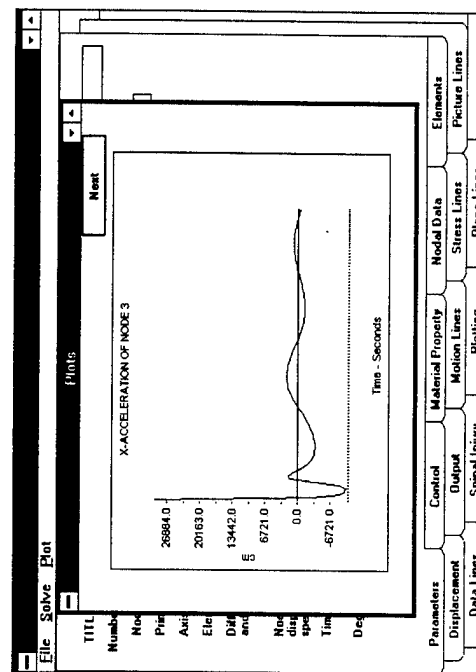
The **Save** option from the **File** menu allows the user to save a variation of the original input file after editing the data. Like the **Open** option, the **Save** option brings up a dialog box that enables the user to select or specify a file name to use in saving the file. The data will be saved in the format which is read by the HSM program. Note that modifying and saving the input data is not enough to run a new HSM simulation -- several files of the model also must be recompiled, as was noted in Section 4.

When the **Solve** option from the menu is selected, the current input file is copied to a file named "hsm.inp" and is then used as input to the HSM program which is run in a DOS shell. After the HSM program completes its execution, the user can select the **Plot** menu option and view the plots generated from the output file. The plot display has a button labeled "Next" which the user can use to move to the next plot. When the last plot is being viewed, the button caption changes to "Done," and when pressed, the plot screen is removed. A screen display of an output file plot is shown in Figure 5-3.

### Figure 5-2 Input File Data Lines



### Figure 5-3 An Example of Plotted Output





## **SECTION 6**

### **VALIDITY OF THE HEAD-SPINE MODEL**

## 6.0 Head-Spine Model Validity.

Questions regarding the validity of a mathematical model naturally arise and should be addressed in the conceptual phase of creating the model. Usually there is some preconceived notion that the model can be validated by comparing the model's predictions to actual data collected on the real world entity being simulated. The true value of a mathematical model lies in its ability to aid in analysis or design of a process or family of processes under study. In fact, in the opinion of the authors, the creation of the model should not be *the* objective at all! Nevertheless, the developers of the HSM stated,<sup>1</sup> "The principal objective of this investigation is the development of a three dimensional, discrete model of the spine and head." Later in their discussion, they implied that the HSM would be used operationally "to investigate the behavior of the spine" in situations "of practical importance."<sup>1</sup> Thus, despite the stated objective, it does appear that the developers of the original HSM recognized and understood the importance of operational validation in establishing the credibility of the model and its use for predictive simulation. Unfortunately, their development plan was not documented in the HSM Technical Reports (TRs),<sup>1-4</sup> nor were their validation studies extensive.

A review of the TRs indicates that the HSM was developed using generally accepted methods employed in the late 1970's and early 1980's. Unfortunately, the HSM and similar models of that time were developed in an academic atmosphere without the benefit of a rigorous development methodology. It appears to BRC that more time and resources were devoted to the actual construction and coding of the model than to its testing and validation. There is extensive documentation of the development of the equations of motion and the physical properties of the spine employed in the HSM. But, there is a relative lack of documentation of verification and validation activities. Some parametric and validation testing was done, but neither the written documentation nor the comments in the HSM code describe all the details of those studies. The available documentation of studies of the HSM cervical spine subsection's response to impulsive acceleration showed encouraging agreement with experimental data.<sup>4</sup> Nevertheless, only a few comparisons were made and the full range of possible head-neck motion was not explored. It is likely that a lack of experimental data or development resources, or both, limited the amount of validation work which was attempted.

BRC's aim was to investigate the feasibility of porting the existing HSM code to the PC environment. BRC has concluded that it is feasible to create a PC-based version of the HSM concept, but not by porting the existing code. The existing HSM program documentation and some of the archived code will be useful in creating a PC version. However, we are not confident that it will be possible to recover the original code and modify it to run in a more general purpose version on any platform without a substantial rewrite of the code. Three key factors, (1) the lack of comprehensive documentation, (2) the code implementation problems noted in Section 4.5, and (3) the limited amount of validation testing combine to substantially lower BRC's confidence in the validity of the archived HSM code. Moreover, in its present form, the HSM is anything but "user friendly." Much of the activity of developing a PC-based version of the HSM would

necessarily be devoted to creating user friendly interfaces. Thus, it is BRC's opinion that the most sensible course to follow would be to completely rewrite the HSM for the PC environment, to conduct that process using a development methodology designed to ensure the model and its limitations are documented, and to employ modern languages and programming constructs to create a model specifically for the PC platform.

There still remains a larger question of whether it is reasonable to expect a meaningful level of validation for the HSM in the future. In its most complex form, the HSM has 42 six degree-of-freedom nodes. This implies it has over 250 possible degrees-of-freedom (DOF). In its actual implementation (and in the real spine), many of the DOF are constrained so that, practically speaking, there are fewer DOF. However, in the model, those constraints are set by literally hundreds of parameters, many of whose values are not known precisely. Thus, the number of unique combinations of parameters makes it impractical to employ conventional parameterization methods with the 42-node version of the HSM. The developers of HSM obviously recognized this problem. Three of the four versions of HSM are simplified so that many of the vertebral nodes are lumped. This dramatically reduces the number of parameter combinations so that fitting the model output to data by parameter adjustment becomes more feasible. Nevertheless, only a few attempts to refine the HSM parameters were described in the project documentation.<sup>1-4</sup>

It appears that the HSM development team assumed that the HSM would be inherently valid if they created a model with a high degree of anatomical fidelity and set its physical properties using the best data obtainable. In fact, this approach has merit if the modeling objective is to understand the relationships between head-spine structure and its motion. It also summarizes what is known and unknown about the physical properties of the head and spine and suggests experiments aimed at their refinement. If, alternatively, the purpose of the model is to recreate (predict) the external motion of the head and spine to potentially injurious acceleration events, there are a number of issues related to validation which must be resolved before the HSM's validation can be established. Validation issues are highlighted in Figure 6-1, and discussed more fully in Paragraphs a-c.

**Figure 6-1 HSM Validation Issues**

- Do mathematically integrated discrete-element models adequately represent naturally integrated structures?
- Are biomechanical properties measured *in vitro* representative of those properties *in vivo*?
- Do biomechanical models of internal structures require validation against force-motion data measured on internal structures *in situ*?
- How does natural redundancy in biological structures affect the validation of mathematical models of that structure?

a. Integrated versus Discrete Nodes. There is a fundamental question related to how the mathematical model resulting from the assembly of models of the discrete elements which comprise the spine can actually simulate the behavior of the ligamentous spine and its associated structures *in vivo*. There is a related issue which can be stated as a question. Are the physical materials properties of individual tissues, obtained *in vitro*, valid when they are combined in the HSM to represented integrated tissues? The present HSM approach is based on the implicit assumption that *in vitro* properties are descriptive of *in vivo* properties and that any and all tissue interactions are adequately described by the equations.

b. Validation with External versus Internal Motion. With present technology, BRC sees little possibility of obtaining dynamic data on internal spinal responses to impulsive acceleration in humans. It may be possible to obtain such data on primate subjects, but as pointed out by the developers of the original model, there are issues relating to scaling primate data for application in human models.<sup>1-4</sup> In any case, primate data will be expensive to acquire and difficult to employ in validation of a human model. Then, the only data which will be readily available will be data on external motions, when what is actually needed to validate the motion of the skeleton *in situ* are internal data. This leads to the next issue.

c. Redundancy in Natural Structures. Natural biological structures are notoriously redundant. This provides “built in” protection against the failure of the entire organism being caused by failure of a single component. The spine is no exception. Spinal position is maintained reflexively by tension in literally hundreds of muscle segments whose actions often oppose each other, as in flexion and extension. This is highly relevant to a model which is based on a high degree



of anatomical fidelity. The hundreds of parameters necessary to mathematically specify the internal position, structure, and function of the spine will inherently contain the same kinds of redundancy as the actual spine. What this boils down to is that external motion of the head and spine will be insensitive to some of the model's parameters. Moreover, adjustments to a particular parameter value can be offset or nullified by adjustments to another. In mathematical terms, the model will be over determined in the context of modeling external motion of the head and spine. This leads to the conclusion that parameters are likely to be intercorrelated and insensitive to changes in the data describing external motion. Ultimately, this means that some parameters can be combined (lumped) or eliminated with little or no effect on the match between the model's predictions and the measured data.

The issues noted above circumscribe and help define the process necessary to create and validate a head-spine model whose predictions of external motions can be validated and subsequently related to internal forces and motions in the skeleton. BRC's recommendations for the development of a PC-based HSM derived from the original model are detailed in Section 7.0 of this report.



**SECTION 7**  
**RECOMMENDATIONS**

## 7.0 Recommendations.

Our research demonstrated that the HSM could be hosted and executed on Personal Computers (PC) under either the Microsoft MS-DOS or Microsoft Windows environment. However, the existing HSM code is largely undocumented and unstructured and was written with older versions of FORTRAN for the mainframe or workstation environment. Therefore, one of the major conclusions of the Phase I research program is that the HSM should be completely re-coded and documented using modern software and numerical routines tailored to the PC environment. Figure 7-1 outlines the technical steps necessary to create a partially validated, user-friendly HSM model for the Windows 95™ operating system. For the purposes of this report, the re-coded HSM will be called "HSM-PC." The following paragraphs discuss each technical step and its purpose.

### 7.1 Plan and Establish the Simulation Validation Protocol.

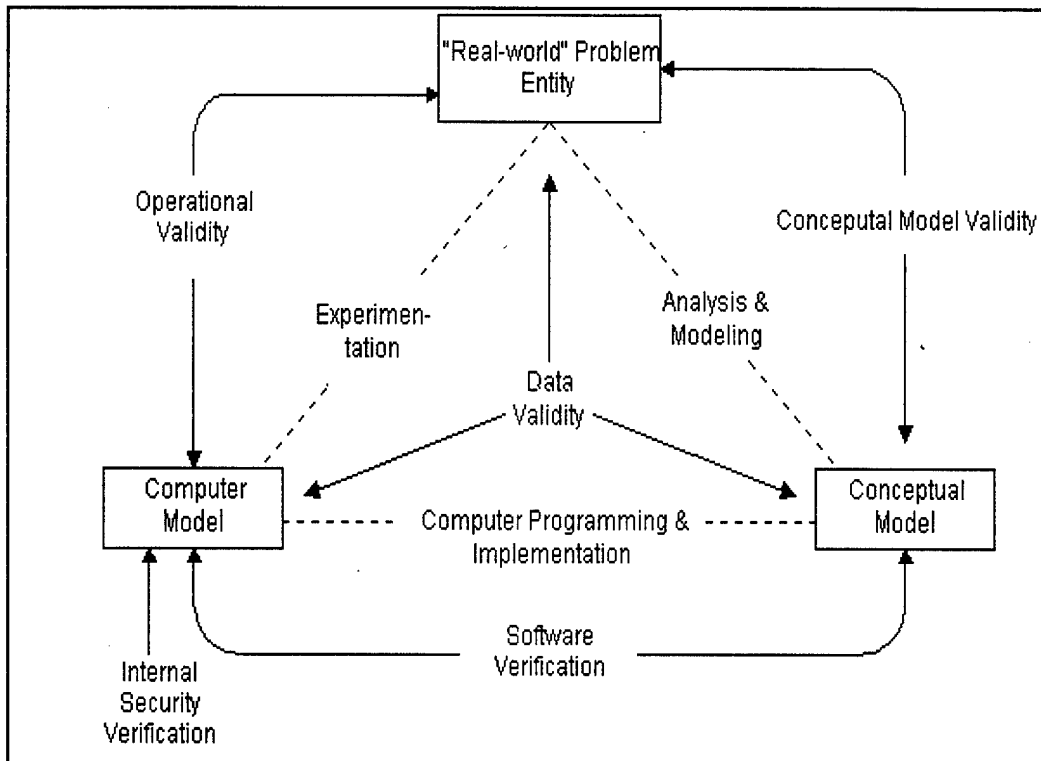
The development of a complex computer simulation such as the HSM-PC demands that a logical and rigorous protocol be employed to manage the risks of creating a software simulation that is difficult to employ and maintain in the "operational environment." The development program should therefore consist of a *planning phase*, during which the development activities are laid out and approved by the sponsors of the program, and an *application phase*, in which the actual development is carried out. BRC recommends that the HSM-PC development program follow a DOD developed methodology described by Knepell and Arango.<sup>1</sup> Their *confidence assessment* methodology provides the necessary controls and procedural framework for controlling the development of a complex simulation, while ensuring the validation of the modeling concept, code verification, documentation, and, most importantly, a systematic evaluation of the model's credibility. Figure 7-2 shows a diagram of the overall scheme. Figure 7-2 was adapted from Figure 2-3 of Reference 1. To explain the method, a few definitions are required.

- a. Conceptual Model Validation. The independent review of the purpose and concept of the model and its implementation to ensure the model addresses the relevant features of the "real-world" entity being simulated--in this case, the head and spine of the human being.
- b. Software Verification. A process by which it is ensured that the software code computes the desired results. Essentially, this process verifies that the coded software implements the desired algorithms correctly and that it performs as desired.

**Figure 7-1 Technical Objectives**

- **Plan and Establish the Simulation Validation Protocol**
- **Create HSM Geometry and Materials Properties Database**
- **Re-Code The HSM for the Personal Computer Environment**
- **Partially Validate the HSM-PC**
- **Conduct Parametric Studies Using the HSM-PC Simulation**
- **Document the HSM-PC and Its Applicability**

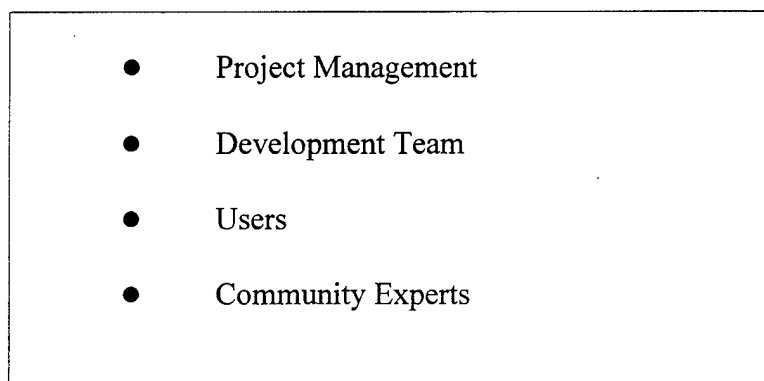
**Figure 7-2 Simulation Validation Process**



- c. Operational Validation. This is the classic validation process necessary of any computer modeling program. Operational validation ensures that the computer simulation has a satisfactory range of accuracy within its intended domain of applicability and that the scope of its domain is defined and documented. In relation to the HSM-PC, operational validation will ensure that the HSM-PC predictions and their errors are understood and that its domain of applicability is known.
- d. Data Validation. The process of documenting the data employed in generating model parameters and equations. For the HSM-PC this will consist of creating, updating, and auditing a database of geometry and materials properties necessary to specify the physical structure and the properties of the constituent tissues of the spine and its associated structures.
- e. Internal Security. Internal security establishes a configuration control procedure and protects the code against tampering or unauthorized modification.

As shown in Figure 7-2, the entire process of *confidence assessment* is designed to ensure the “real-world problem entity.” In this case, the human head and spine are represented and simulated in the computer model as logically and accurately as possible. The entire process is managed by an Assessment Team whose members are listed in Figure 7-3.

**Figure 7-3 Assessment Team Representatives**



The roles of the Assessment Team members are briefly described below.

- a. Management. A member from program Management will be designated to serve as an informed member of the Assessment Team. The role of management in the process is to provide support and direction as necessary to ensure sufficient resources are available to complete the development program on schedule and within budget.

- b. Development. Developers include the programmers, consultants, scientists, and computer infrastructure experts necessary to implement the development plan for the HSM-PC development.
- c. Users. The users group represents knowledgeable scientists and practitioners who possess the necessary skill and technical knowledge to employ the model in an investigative or problem solving environment. Members of this group should not come from the Development community.
- d. Community Experts. To provide an objective review of the HSM-PC and its development process, subject matter experts (SME's) will be employed as members of the review team.

The model evaluation approach is detailed in Simulation Validation.<sup>1</sup> This process is too extensive to be completely described here. Briefly, it provides: (1) the detailed methodology and tools to continuously assess, verify, validate, and document a complex software simulation during its development; (2) an explicit method to assess the credibility of the model's simulations and its domain of applicability; and (3) an audit trail to support the confidence assessment process governed by the Assessment Team. Although the confidence assessment methodology will add both time and expense to the HSM-PC development effort, it will ensure that the HSM-PC software is properly assessed and documented.

## **7.2 Create a Database of the Geometry and Materials Properties Data Contained in the AAMRL HSM.**

The researchers who created the original AAMRL HSM<sup>1-4</sup> created an extensive collection of anatomic, geometric, and materials property data relating to the spine and is associated tissues. These data were documented in the program Technical Reports or in the HSM code. BRC reviewed the geometry and materials properties data related to the HSM and found them to be generally complete. However, no audit was conducted to confirm that the data contained in the HSM documentation was actually the best available.

As noted above, the original HSM data is somewhat dated. A survey of the more recent literature should be conducted to support a systematic update and audit of the physical geometry and biomechanical properties of the head, spine, and associated tissues. Once the update and audit are complete, the geometry and materials properties data should be loaded in a "Properties Database" for on-line access by the HSM-PC. This database should be implemented in an environment which will be independently accessible for other purposes as a stand alone database.

### **7.3 Create User Friendly Interfaces**

As shown in Figure 7-4, the HSM software uses an input file in card image format which is tedious to assemble and difficult to decode. The user is required to encode the initial state of the model as a precise sequence of control and data records. The program output is a large text file which includes graphical depictions of the model states. To aid review of the output, the user has access to a set of drawing commands which allows some flexibility in viewing the results of the analysis. The Phase I effort added a new prototype user interface which allows the user to view, edit and save input files. The screen pages provide labels for the input data and in some cases choices of possible field values. This Phase I interface also reads the output file generated by the analysis and displays X-Y plots on the screen based on the variable values computed by the analysis.

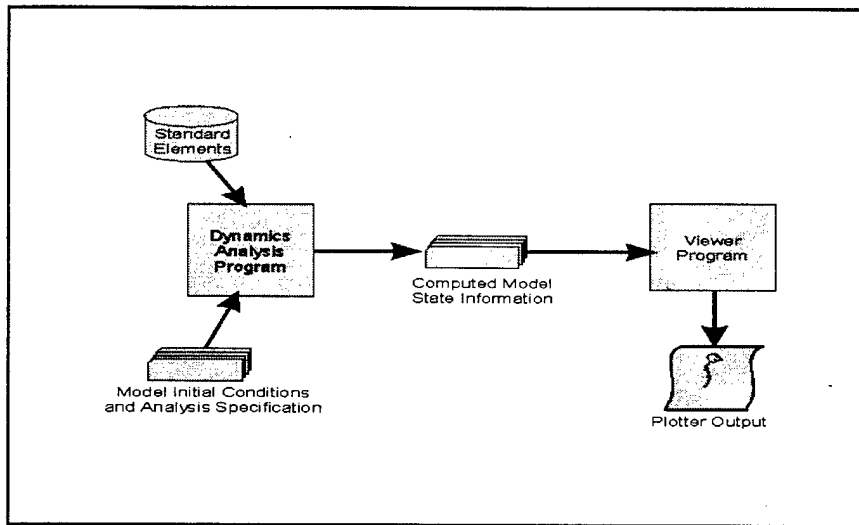
Both the input and output GUIs for the HSM-PC should be studied and tailored to the PC environment. Of particular importance are provisions to manage the large numbers of input specifications necessary to specify the initial conditions and forcing functions for the HSM-PC. The GUI should display the initial position of the head and spine and its supporting structures as well as allow display of the forcing function. The use of the Materials Properties Database will facilitate preserving and setting parameters which are not changed frequently. The output GUI should be able to display "snapshots" of two- and three-dimensional views of the position of the head and spine as a function of time. The user should be able to easily create graphical displays of the position, velocity, and acceleration of particular nodes versus time or displays of the kinematic variables versus each other and the amplitude of the forcing function.

### **7.4 Re-Code the HSM for the PC Environment.**

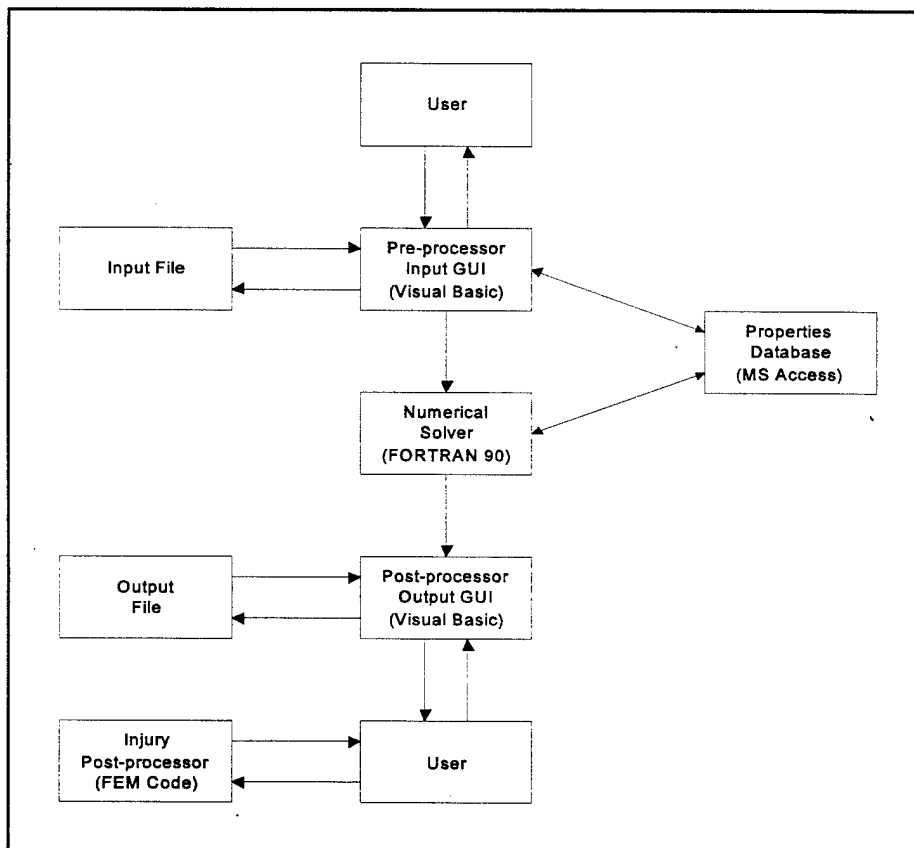
It is suggested that a "build-up" approach be employed to ensure the proper verification and documentation of the software code. Figure 7-5 illustrates the suggested program architecture. It is recommended that the HSM-PC be hosted on IBM compatible PC's running the Windows 95™ operating system. This will ensure its compatibility with the 32-bit environment and dictate that a more capable machine be employed to execute the code. The Input/Output GUI's should be programmed in Microsoft Visual Basic™ and/or Microsoft Visual C++™ and the numerical analysis and solver modules should be written in a language which emphasizes speed of execution such as Microsoft's Fortran-90™. The development team should consider the use of computer assisted software engineering (CASE) tools such as SD/FAST. SD/FAST is a multibody dynamics modeling application for the PC which generates the Kane's Equations<sup>10</sup> (differential equations) as Fortran source code subroutines which can be incorporated directly in independently developed Fortran code. The geometry and materials properties database will be implemented in a compatible database application such as Microsoft Access™. All of the software applications should employ the Object Oriented Programming (OOP) technology. The choice of a single software manufacturer and the OOP programming constructs will ensure that there is mutual compatibility between the operating system and the major programming applications employed to create the HSM-PC.



**Figure 7-4 Data Flow of Original HSM**



**Figure 7-5 Overall Program Structure**



The use of a build up approach to create the model will ensure that the development team develops the appropriate discipline demanded by the CA process early in the program. It is suggested that the sequence of creating subsegment models follow those outlined in Figure 7-6, which is a roadmap of the suggested development process. As shown, a full, 3-D head-cervical spine model should be developed and partially validated to prove the viability of the development process. The head-cervical spine model will be a useful tool in its own right and independent evaluation and further validation could begin on that model subsection before the entire development process is complete.

## **7.5 Validation of the HSM-PC Simulation.**

Operational validation of the simulations produced by subsystem modules of the HSM-PC should commence as soon as the software subsystem modules are created and coded. Predictive validation against experimental data or known principles of physics should take place where possible.

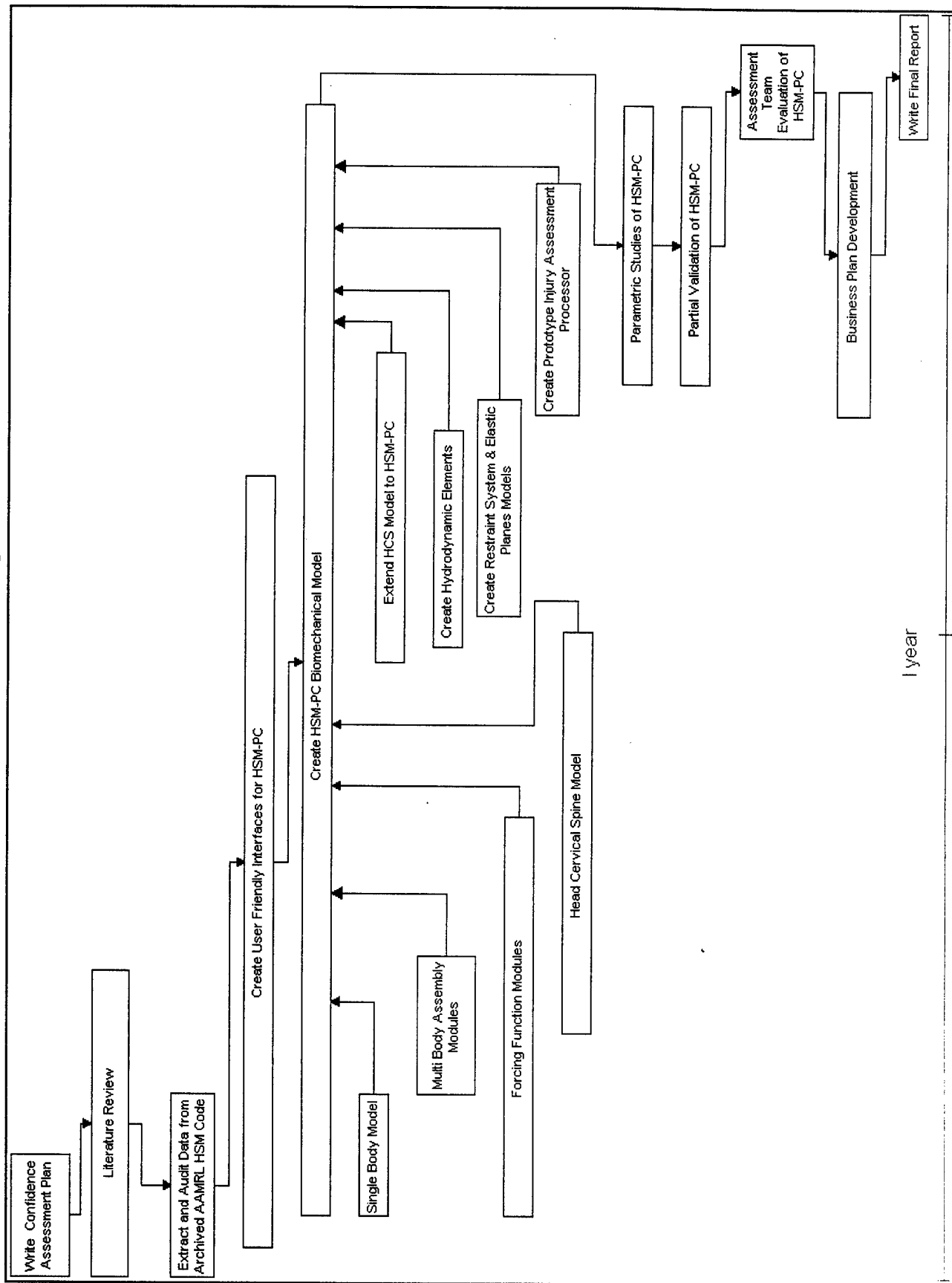
Because the HSM-PC will be complex and will contain many degrees of freedom, the scope of its predictions will be enormous. It will, therefore, be virtually impossible to validate every possible prediction of the HSM-PC. Nevertheless, an attempt should be made to employ suitable data where feasible to define the scope of the HSM-PC applicability. Within those domains where the use of the HSM-PC is verified and validated, data on its accuracy should be sought to allow the user to gauge the utility of the HSM-PC simulation predictions in relation to other analytic tools or actual data.

## **7.6 Conduct Parametric Studies with the HSM-PC.**

Parametric Studies should be conducted to examine the sensitivity of the HSM-PC simulation to changes in parameter values. Parametric investigations will reveal which parameters must be set relatively precisely to ensure the HSM-PC simulations are representative of the dynamics of the human spine. Parameter sensitivities and covariance issues should be addressed as early as possible in the process so that redundant or extraneous parameters can be eliminated from the model. This process will ensure parsimony in model parameters and a verified and validated HSM-PC.

In the process required to "fit" the HSM-PC to experimental data, parameters should be varied and the response of the models compared to experimental data until "best fits in the least squares sense"<sup>11</sup> are obtained. However, as noted above, it is to be expected that a model with many degrees of freedom will be somewhat over determined, i.e., there will be interdependence and covariance between the model parameters. The interdependence between parameters and their covariance should be studied, perhaps using classic parametric sensitivity study methods such as Monte-Carlo.<sup>12-15</sup> Another approach might be to examine the model residuals (the difference between predicted and actual motion) for their structure to define "how much" of the actual motion is/or is not simulated by the model. The relationships between the residuals and the forcing function amplitude and its frequency content should also be investigated. The iterative reduction of parameter

**Figure 7-6 Roadmap to Development Process**



correlation will likely be a tedious process and will be very demanding on computer resources. It will require putting the Numerical Solver inside a non-linear optimization structure and repeatedly finding best fit solutions as the parameters are varied. If this process is started early in the development process, an intuitive understanding of parameter sensitivities will likely evolve which will serve the development team as the development proceeds. Parameters whose values dominate the model response characteristics and those which produce little or no change in the model's response or its residuals should be identified and studied in detail. Parameters to which the model's response is insensitive should be flagged as candidates for elimination from the final model and their effects should be tracked as the model's complexity increases.

#### **7.7 Document the HSM-PC and Its Applicability.**

The confidence assessment (CA) methodology described in Reference 1 ensures that the basic material from which the final documentation is drawn is developed in a complete and systematic manner. The CA process is iterative. It should be applied at the subsystem level as the code for major subsystems is developed. After each evaluation, a report should be issued which documents the findings of the subsystem CA.

#### **7.8 Conclusion.**

Biodynamic Research Corporation (BRC) of San Antonio, TX, completed an SBIR Phase I project to port the Air Force's Head-Spine Model (HSM) to a PC-DOS environment and provide a recommended roadmap for the future of the HSM. The impetus for this project was the Air Force's desire to have a software tool capable of modeling the internal forces and motions of the human head and spine during impulsive acceleration events.

Although the code transfer was successful, BRC discovered several "problems" with the Head-Spine Model which made creating a general purpose HSM code impossible. It is BRC's belief that the material and geometry data of the HSM model, as well as some of the model algorithms and logic, can best be used by recreating the model in an object-oriented software language such as C++ or Fortran 90. The cost and effort required to understand the current version, debug coding and algorithm errors, and document the code is far greater than simply extracting the useful data and starting over. Therefore, BRC recommends that the HSM be rewritten for the PC environment and that the development program be conducted under the confidence assessment process outlined above.

## **SECTION 8**

### **REFERENCES**



## 8.0 References.

1. Belytschko T., Schwer L., and Schultz A. A Model for Analytic Investigation of Three-Dimensional Head-Spine Dynamics. AMRL-TR-76-10; 1976.
2. Belytschko T. and Privityzer E. Refinement and Validation of a Three-Dimensional Head-Spine Model. AMRL-TR-78-7; 1978.
3. Williams T. and Belytschko T. A Dynamic Model of the Cervical Spine and Head. AFAMRL-TR-81-5; 1981.
4. Belytschko T., Williams J., and Rencis M. Head-Spine Structure Modeling: Enhancements to Secondary Loading Path Model and Validation of Head-Cervical Spine Model. AAMRL-TR-85-019; 1985.
6. Dietrich, M., Kedzior, K., and Zagrajek, T. A Biomechanical Model of the Human Spinal System. Proceedings from the Institution of Mechanical Engineers Vol. 205; 1991.
7. Helleur, C., Gracovetsky, S., Farfan, H. Tolerance of the Human Cervical Spine to High Acceleration: A Modelling Approach. Aviation, Space, and Environmental Medicine; October, 1984.
8. McNally, D.S. and Arridge, R.G.C. An Analytical Model of Intervertebral Disc Mechanics. Journal of Biomechanics, Vol. 28, No. 1, pp. 53-68m 1995.
9. Farfan, H., Gracovetsky, S., and Helleur, C. Cervical Spine Analysis for Ejection Injury Prediction. AFOSR-TR-83-0590. (1982).
10. Hollars, M.G., et al. SD/FAST User's Manual. Symbolic Dynamics, Inc. Mountain View, CA (1991).
11. Press, W.H., et al. Numerical Recipes, *The Art of Scientific Computing*. Cambridge University Press, New York, NY (1986).
12. Rubinstein, R.Y. Simulation and the Monte Carlo Method. John Wiley and Sons. New York, NY (1981).
13. Bosch, P.P.J. van den and Klauw, A.D. van der. Modeling, Identification, and Simulation of Dynamical Systems. CRC Press. Boca Raton, FL (1994).
14. Ripley, B.D. Stochastic Simulation. John Wiley and Sons. New York, NY (1987).
15. Thompson, J.R. Empirical Model Building. John Wiley and Sons. New York, NY (1989).

Other references not cited.

16. Allen, L. Finite Element Analysis of the Visco-Elastic Interaction of the Annulus Fibrosis and Nucleus Pulposus Within the Human Intervertebral Joint. (1981)
17. Perry, C.E., Bonetti, D.M., Brinkley, J.W. The Effect of Variable Seat Back angles on Human Response to +Gz Impact Accelerations. AL-TR-1991-0110 (1991)
18. Burns, M.L. Analytical Modeling of Load-Deflection Behavior of Intervertebral Discs Subjected to Axial Compression by Exact Parametric Solutions of Kelvin-Solid Models. AFOSR-TR-81-0653 (1980)



## **APPENDIX A**

### **HSM CODE FOR A PC-DOS COMPUTER**

```

SUBROUTINE ALPHAP (NUMNP,XC,YC,ZC,EULCO,UD,INMESH)
C
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
Cbrc
Cbrc REPLACED COMMONS WITH INCLUDES AND ELIMINATED GO TO'S - DJP
Cbrc
  INCLUDE 'ADJUST.COM'

  COMMON /PLOT/ PLTD CD,FSTPLT(9),PELVIS,IDPN(10),NPPTS(10),IDP(10),
1    INT,PRNT
  INTEGER PLTD CD, PRNT
  DIMENSION EULCO(3,3,NBLAMB), XC(NXC), YC(NYC), ZC(NZC), UD(NX1),
1    INMESH(NINMEH)
C
  INTEGER PELVNN
  INTEGER T1,SPNN,BPNN,TPNN,HEADNN
  INTEGER C7,C2
  INTEGER STERNM,RIBBGN,RIBEND,BPR,BPL
  DATA ITYPE/4HBODY/
  DATA CERV/3HCYN/
  IPTC=0
  PELVNN = INMESH(231)
  L5=INMESH(232)
C
C.... L5=INMESH(248)
C
  T1=INMESH(248)
  HEADNN=INMESH(249)
  NNSP=16
  NNBP=15
  NNTP=21
  SPNN=INMESH(NNSP)
  BPNN=INMESH(NNBP)
  TPNN=INMESH(NNTP)
C
C.... BPNN=INMESH(223)
C.... TPNN=INMESH(229)
C.... SPNN=INMESH(224)
C
  NDBP=6*(BPNN-1)
  XOFF=XC(BPNN)+UD(NDBP+1)
  YOFF=YC(BPNN)+UD(NDBP+2)
  ZOFF=ZC(BPNN)+UD(NDBP+3)
C
  IF ( PELVIS.EQ. ' ' ) THEN
    NBPPELV=INMESH(7)
    NDPELV = 6*(NBPPELV-1)
    X = XC(NBPPELV)+UD(NDPELV+1)-XOFF
    Y = YC(NBPPELV)+UD(NDPELV+2)-YOFF
    Z = ZC(NBPPELV)+UD(NDPELV+3)-ZOFF
    NODE = 1
C
  IF ( PRNT.EQ. 0 )
1    WRITE (6,904) NODE , IPTC , ITYPE , X , Y , Z,
2      (( EULCO(M,L,PELVNN), M=1,3), L=1,3,2)
  WRITE (7,903) NODE , IPTC , ITYPE , X , Y , Z,
1    (( EULCO(M,L,PELVNN), M=1,3), L=1,3,2)
  END IF
C
C
DO KNODE=L5,T1
  NDSP=6*(SPNN-1)
  NDBP=6*(BPNN-1)
  NDTP=6*(TPNN-1)
C
  V3X=XC(TPNN)-XC(BPNN)+UD(NDTP+1)-UD(NDBP+1)
  V3Y=YC(TPNN)-YC(BPNN)+UD(NDTP+2)-UD(NDBP+2)
  V3Z=ZC(TPNN)-ZC(BPNN)+UD(NDTP+3)-UD(NDBP+3)
C
  V2X=XC(SPNN)-XC(BPNN)+UD(NDSP+1)-UD(NDBP+1)
  V2Y=YC(SPNN)-YC(BPNN)+UD(NDSP+2)-UD(NDBP+2)
  V2Z=ZC(SPNN)-ZC(BPNN)+UD(NDSP+3)-UD(NDBP+3)
C
  V1X=V2Y*V3Z-V3Y*V2Z
  V1Y=V2Z*V3X-V3Z*V2X

```

```

V1Z=V2X*V3Y-V3X*V2Y
C
V1L=DSQRT(V1X*V1X+V1Y*V1Y+V1Z*V1Z)
C
V1X=V1X/V1L
V1Y=V1Y/V1L
V1Z=V1Z/V1L
C
V3L=DSQRT(V3X*V3X+V3Y*V3Y+V3Z*V3Z)
C
V3X=V3X/V3L
V3Y=V3Y/V3L
V3Z=V3Z/V3L
C
X=XC(BPNN)+UD(NDBP+1)-XOFF
Y=YC(BPNN)+UD(NDBP+2)-YOFF
Z=ZC(BPNN)+UD(NDBP+3)-ZOFF
C
NODE=KNODE-L5+2
C
IF ( PRNT .EQ. 0 )
1 WRITE (6,904) NODE,IPTC,ITYPE,X,Y,Z,V1X,V1Y,V1Z,V3X,V3Y,V3Z
WRITE (7,903) NODE,IPTC,ITYPE,X,Y,Z,V1X,V1Y,V1Z,V3X,V3Y,V3Z
C
NNSP=NNSP+13
NNBP=NNBP+13
NNTP=NNTP+13
SPNN=INMESH(NNSP)
BPNN=INMESH(NNBP)
TPNN=INMESH(NNTP)
C
END DO
C
C
IF (HEADNN.NE.19 .OR. HEADNN.NE.21) THEN
C7=INMESH(340)
C2=INMESH(345)
SPNN=INMESH(268)
BPNN=INMESH(256)
TPNN=INMESH(257)
NODE=0
C
DO KNODE=C7,C2
NDSP=6*(SPNN-1)
NDBP=6*(BPNN-1)
NDTP=6*(TPNN-1)
C
V3X=XC(TPNN)-XC(BPNN)+UD(NDTP+1)-UD(NDBP+1)
V3Y=YC(TPNN)-YC(BPNN)+UD(NDTP+2)-UD(NDBP+2)
V3Z=ZC(TPNN)-ZC(BPNN)+UD(NDTP+3)-UD(NDBP+3)
C
V2X=XC(SPNN)-XC(BPNN)+UD(NDSP+1)-UD(NDBP+1)
V2Y=YC(SPNN)-YC(BPNN)+UD(NDSP+2)-UD(NDBP+2)
V2Z=ZC(SPNN)-ZC(BPNN)+UD(NDSP+3)-UD(NDBP+3)
C
V1X=V2Y*V3Z-V3Y*V2Z
V1Y=V2Z*V3X-V3Z*V2X
V1Z=V2X*V3Y-V3X*V2Y
C
V1L=DSQRT(V1X*V1X+V1Y*V1Y+V1Z*V1Z)
C
V1X=V1X/V1L
V1Y=V1Y/V1L
V1Z=V1Z/V1L
C
V3L=DSQRT(V3X*V3X+V3Y*V3Y+V3Z*V3Z)
C
V3X=V3X/V3L
V3Y=V3Y/V3L
V3Z=V3Z/V3L
C
X=XC(BPNN)+UD(NDBP+1)-XOFF
Y=YC(BPNN)+UD(NDBP+2)-YOFF
Z=ZC(BPNN)+UD(NDBP+3)-ZOFF
C
NODE=NODE+1

```

```

      IPTC=32
      IF (NODE.EQ.1) IPTC=20
C
      IF ( PRNT .EQ. 0 )
1  WRITE (6,904) NODE,IPTC,CERV,X,Y,Z,V1X,V1Y,V1Z,V3X,V3Y,V3Z
  WRITE (7,903) NODE,IPTC,CERV,X,Y,Z,V1X,V1Y,V1Z,V3X,V3Y,V3Z
C
      SPNN=SPNN+1
      BPNN=BPNN+2
      TPNN=TPNN+2
C
      END DO
      END IF
C
      IF (NUMNP.GE.471) THEN
        STERNM=INMESH(471)
        RIBBGN=INMESH(451)
        RIBEND=INMESH(469)
        NNBPR=296
        NNBPL=303
        BPR=INMESH(NNBPR)
        BPL=INMESH(NNBPL)
        IPTC=0
C
        IF (NUMNP.LT.STERNM) RETURN
C
        DO KNODE=RIBBGN,RIBEND,2
          NDBP=6*(BPR-1)
          X=XC(BPR)+UD(NDBP+1)-XOFF
          Y=YC(BPR)+UD(NDBP+2)-YOFF
          Z=ZC(BPR)+UD(NDBP+3)-ZOFF
C
          NODE=KNODE-RIBBGN+19
C
          IF ( PRNT .EQ. 0 )
1  WRITE (6,904) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,KNODE),
2  M=1,3),L=1,3,2)
        WRITE (7,903) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,KNODE),M=1,3),
1  L=1,3,2)
C
          NDBP=6*(BPL-1)
          X=XC(BPL)+UD(NDBP+1)-XOFF
          Y=YC(BPL)+UD(NDBP+2)-YOFF
          Z=ZC(BPL)+UD(NDBP+3)-ZOFF
C
          INODE=KNODE+1
          NODE=INODE-RIBBGN+19
C
          IF ( PRNT .EQ. 0 )
1  WRITE (6,904) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,INODE),
2  M=1,3),L=1,3,2)
        WRITE (7,903) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,INODE),M=1,3),
1  L=1,3,2)
C
          NNBPR=NNBPR+14
          NNBPL=NNBPL+14
          BPL=INMESH(NNBPL)
          BPR=INMESH(NNBPR)
C
          END DO
C
          NDSTER=6*(STERNM-1)
          X=XC(STERNM)+UD(NDSTER+1)-XOFF
          Y=YC(STERNM)+UD(NDSTER+2)-YOFF
          Z=ZC(STERNM)+UD(NDSTER+3)-ZOFF
C
          NODE=STERNM-RIBBGN+19
C
          IF ( PRNT .EQ. 0 )
1  WRITE (6,904) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,STERNM),
2  M=1,3),L=1,3,2)
        WRITE (7,903) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,STERNM),M=1,3),
1  L=1,3,2)
          END IF
C
          NDHEAD = 6 * ( HEADNN-1 )

```

```

X=XC(HEADNN)+UD(NDHEAD+1)-XOFF
Y=YC(HEADNN)+UD(NDHEAD+2)-YOFF
Z=ZC(HEADNN)+UD(NDHEAD+3)-ZOFF
C
  NODE=40
C
  IF ( PRNT.EQ. 0 )
1  WRITE (6,904) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,HEADNN),M=1,3),
2      L=1,3,2)
  WRITE (7,903) NODE,IPTC,ITYPE,X,Y,Z,((EULCO(M,L,HEADNN),M=1,3),
1      L=1,3,2)
C
C
  RETURN
C
903 FORMAT (I2,I2,A4,3F8.3,6F8.6)
904 FORMAT (1X,I2,I2,A4,3F8.3,6F8.6)
  END

```

```

SUBROUTINE ASSBLE (IX,XC,YC,ZC,E,SMASS,RMASS,DICOS,INDEX,STRS,IPT,
IAL)
C
Cbrc
Cbrc REPLACED COMMONS WITH INCLUDES AND MOST GOTOS WITH BLOCK IF AND DO'S
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
INCLUDE 'ADJUST.COM'
INCLUDE 'SIZE.COM'

COMMON /NEIGEN/ NAM,NR
DIMENSION IX(14,NIX), XC(NXC), YC(NYC), ZC(NZC), E(12,NE),
1   SMASS(NSMASS), RMASS(3,3,NBLAMB), EULCO(3,3),
2   DICOS(3,3,NDICOS), INDEX(NINDEX), STRS(NSTRS),
3   IPT(NIPT), AL(NAL)
INCLUDE 'CONTRL.COM'
INCLUDE 'NUMINT.COM'

IF(KONTRL(3).EQ.2) WRITE(6,201)

MEQ=NNODE*NDGREE
NEQ=NPRI*NDGREE
ISEC=0
MAXIPT=1

IF (NUMSEC.NE.0) THEN
  READ (5,901) (IPT(LEN),LEN=1,NUMSEC)
  DO LEN=1,NUMSEC
    IF (MAXIPT.LT.IPT(LEN)) MAXIPT=IPT(LEN)
  END DO
END IF

DO I=1,NPRI
  NN=(I-1)*NDGREE
  DO M=1,3
    LEN=NN+3
    RMASS(M,M,I)=SMASS(LEN+M)
  END DO
END DO

IADD=1
INDEX(1)=1
DO JE=1,NELE
  NOPT=IX(10,JE)
  ISEC=IX(11,JE)
  IF (NOPT.EQ.1) IADD=16
  IF (NOPT.EQ.3) IADD=46
  IF (NOPT.EQ.4) IADD=40
  IF (NOPT.EQ.5) IADD=48+21*ISEC
  IF (NOPT.EQ.6) IADD=5
  IF (NOPT.EQ.2) THEN
    I1=1
    IF (ISEC.NE.0) I1=IPT(ISEC)
    IADD=41+6*I1
    IF (KONTRL(5).GT.0) IADD=41+8*I1
  END IF
  INDEX(JE+1)=INDEX(JE)+IADD
  IND=INDEX(JE)-1
  IF (NOPT.EQ.2 .OR. ISEC.NE.0) THEN
    MTYP=IX(9,JE)
    KK=IADD+IND
    I2=2*I1
    IF (KONTRL(5).GT.0) KK=KK-I2
    DO M=1,I2
      STRS(KK)=E(3,MTYP)
      KK=KK-1
    END DO
  END IF
  IF (NOPT.EQ.5) THEN
    ID=INDEX(JE)+27
    IF (MAXIPT.LT.ISEC) MAXIPT=ISEC
    NIND = NSTRS-IND
  ELSE IF (NOPT.NE.6) THEN
    CALL BASME (JE,NDGREE,IX,XC,YC,ZC,E,SMASS,DICOS,RMASS,AL)
  ELSE

```

```

        CYCLE
      END IF
    END DO

    DO I=1,NPRI
      NN=(2*I-1)*3
      DO MJ=1,3
        DO JM=1,3
          EULCO(MJ,JM)=0.D0
        END DO
      END DO
      IF (RMASS(1,1,I).EQ.0.) CYCLE
      IF (KONTRL(3).NE.0) THEN
        IF (KONTRL(3).EQ.2) GO TO 202
        EULCO(1,1)=1.D0
        EULCO(2,2)=1.D0
        EULCO(3,3)=1.D0
        SMASS(NN+1)=RMASS(1,1,I)
        SMASS(NN+2)=RMASS(2,2,I)
        SMASS(NN+3)=RMASS(3,3,I)
        GO TO 14
      END IF
      RMASS(2,1,I)=RMASS(1,2,I)
      RMASS(3,1,I)=RMASS(2,2,I)
      DO MJ=1,3
        RMASS(MJ,2,I)=RMASS(MJ,3,I)
      END DO
    C
    NAM = 3 + (3*3 - 3)/2
    NR = 3*3
    CALL EIGEN (RMASS(1,1,I),EULCO,3,0)
  C
  C.... NORMALIZE AND DETERMINE THIRD EIGENVECTOR FOR RIGHT HAND SYSTEM
  C
  DO NE=1,2
    ENORM=DSQRT(EULCO(1,NE)*EULCO(1,NE)+EULCO(2,NE)*EULCO(2,NE)+
1      EULCO(3,NE)*EULCO(3,NE))
    EULCO(1,NE)=EULCO(1,NE)/ENORM
    EULCO(2,NE)=EULCO(2,NE)/ENORM
    EULCO(3,NE)=EULCO(3,NE)/ENORM
  END DO
  EULCO(1,3)=EULCO(2,1)*EULCO(3,2)-EULCO(2,2)*EULCO(3,1)
  EULCO(2,3)=EULCO(3,1)*EULCO(1,2)-EULCO(3,2)*EULCO(1,1)
  EULCO(3,3)=EULCO(1,1)*EULCO(2,2)-EULCO(1,2)*EULCO(2,1)
  C
  C.... AVERAGE INERTIAS
  C
  AMASS=(RMASS(1,1,I)+RMASS(3,1,I)+RMASS(3,2,I))/3.D0
  SMASS(NN+1)=AMASS
  SMASS(NN+2)=AMASS
  SMASS(NN+3)=AMASS
  C
  C.... IF(NNODE.EQ.11) GO TO 210
  C
  SMASS(NN+1)=RMASS(1,1,I)
  SMASS(NN+2)=RMASS(3,1,I)
  SMASS(NN+3)=RMASS(3,2,I)
  GO TO 14
  C
  C READ-IN X-BAR, Y-BAR VECTORS.
  C
  202 READ(5,203)NODE,EULCO(1,1),EULCO(2,1),EULCO(3,1),
1    EULCO(1,2),EULCO(2,2),EULCO(3,2)
  WRITE(6,204) NODE
  C
  C NORMALIZE X-BAR, Y-BAR VECTORS.
  C
  DO NE=1,2
    ENORM=DSQRT(EULCO(1,NE)*EULCO(1,NE)+EULCO(2,NE)*EULCO(2,NE)+
1      +EULCO(3,NE)*EULCO(3,NE))
    EULCO(1,NE)=EULCO(1,NE)/ENORM
    EULCO(2,NE)=EULCO(2,NE)/ENORM
    EULCO(3,NE)=EULCO(3,NE)/ENORM
  END DO
  C
  C FIND Z-BAR UNIT VECTOR FROM CROSS-PRODUCT OF X-BAR AND Y-BAR.

```

```

C
EULCO(1,3)=EULCO(2,1)*EULCO(3,2)-EULCO(2,2)*EULCO(3,1)
EULCO(2,3)=EULCO(3,1)*EULCO(1,2)-EULCO(3,2)*EULCO(1,1)
EULCO(3,3)=EULCO(1,1)*EULCO(2,2)-EULCO(1,2)*EULCO(2,1)
DO NE=1,3
  WRITE(6,207)EULCO(NE,1),EULCO(NE,2),EULCO(NE,3)
END DO

C
14 DO MJ=1,3
  DO JM=1,3
    RMASS(MJ,JM,1)=EULCO(MJ,JM)
  END DO
END DO
END DO

WRITE (6,902)
WRITE (6,903) (SMASS(LS),LS=1,NEQ)

C
C.... CALCULATE R-ZERO AND R-ZERO-BAR FOR ALL ELEMENTS AND
C.... STORE IN STRS ARRAY
C
DO I=1,NELE
  MJ=INDEX(I)-1
  NOPT=IX(10,I)
  LOOP1=2
  LOOP2=2
  IPADD=2
  IF (NOPT.EQ.5) THEN
    LOOP1=3
    LOOP2=1
    IPADD=3
  END IF
  J3=4
  DO J1=1,LOOP1
    IF (NOPT.EQ.4) GO TO 21
    K=IX(J1,I)
    KK=IX(J1+IPADD,I)

C
C.... SKIP BODY COMPONENT CALCULATION FOR SPRING ELEMENTS (TYPE=1)
C
    IF (NOPT.EQ.1) GO TO 23
    DO L=1,LOOP2
      DO M=1,3
        SUM=0.D0
        MJ=MJ+1
        KEV=L
        IF (NOPT.EQ.5) KEV=3
        DO N=1,3
          SUM=SUM+RMASS(N,M,KK)*DICOS(N,KEV,I)
        END DO
        STRS(MJ)=SUM
      END DO
    END DO
    GO TO 23

C
C.... LOOP FOR R-ZERO AND R-ZERO-BAR FOR PVP ELEMENT(TYPE=4)
C
21 J2=J1-1
  J3=0
  J4=3*J2
22 J3=J3+1
  K=IX(J3+J4,I)
  KK=IX(7+J2,I)

C
C.... CALCULATE R-ZERO
C
23 STRS(MJ+1)=XC(K)-XC(KK)
  STRS(MJ+2)=YC(K)-YC(KK)
  STRS(MJ+3)=ZC(K)-ZC(KK)
  MK=MJ
  MJ=MJ+3

C
C.... CALCULATE R-ZERO-BAR
C
DO NN=1,3
  MJ=MJ+1

```



```

      STRS(MJ)=RMASS(1,NN,KK)*STRS(MK+1)+RMASS(2,NN,KK)*STRS(MK+2)
1      +RMASS(3,NN,KK)*STRS(MK+3)
      END DO
      IF (J3.LT.3) GO TO 22
      END DO
      END DO
      RETURN
C
201 FORMAT('0 INITIAL BODY UNIT VECTOR COMPONENTS',/,
1      ' AT NODE')
203 FORMAT(15,5X,6F10.4)
204 FORMAT(' ',5X,I5)
207 FORMAT(' ',9X,3F10.4)
901 FORMAT (16I5)
902 FORMAT (/5X,37HTOTAL PRIMARY NODE MASS ARRAY  SMASS/)
903 FORMAT (2X,1P6D20.4)
      END

```

```

SUBROUTINE BASME (I,NDGREE,IX,XC,YC,ZC,E,SMASS,DICOS,RMASS,AL)
C
Cbrc
Cbrc CHANGED COMMON TO INCLUDE AND GOTO'S TO BLOCK IF AND DO
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
INCLUDE 'ADJUST.COM'
DIMENSION XC(NXC),YC(NYC),ZC(NZC),E(12,NE),IX(14,NTX),
1 SMASS(NSMASS)
DIMENSION DICOS(3,3,NDICOS), AL(NAL), RMASS(3,3,NBLAMB)
C
C.... CHECK FOR PVP ELEMENTS (ETYP=4)
C
IF (IX(10,I).NE.4) THEN
N1=IX(1,I)
N2=IX(2,I)
N3=IX(3,I)
N4=IX(4,I)
XX=XC(N2)-XC(N1)
YY=YC(N2)-YC(N1)
ZZ=ZC(N2)-ZC(N1)
AL(I)=DSQRT(XX*XX+YY*YY+ZZ*ZZ)
C
C.... SKIP DIRECTION COSINE CALCULATION FOR SPRING ELEMENTS
C
IF (IX(10,I).NE.1) THEN
DICOS(1,1,I)=XX/AL(I)
DICOS(2,1,I)=YY/AL(I)
DICOS(3,1,I)=ZZ/AL(I)
LOCNOD=IX(8,I)
VAX=XC(LOCNOD)-XC(N1)
VAY=YC(LOCNOD)-YC(N1)
VAZ=ZC(LOCNOD)-ZC(N1)
A=DSQRT(VAX*VAX+VAY*VAY+VAZ*VAZ)
AX=VAX/A
AY=VAY/A
AZ=VAZ/A
E3X=AZ*DICOS(2,1,I)-AY*DICOS(3,1,I)
E3Y=AX*DICOS(3,1,I)-AZ*DICOS(1,1,I)
E3Z=AY*DICOS(1,1,I)-AX*DICOS(2,1,I)
ANORM=DSQRT(E3X*E3X+E3Y*E3Y+E3Z*E3Z)
DICOS(1,3,I)=E3X/ANORM
DICOS(2,3,I)=E3Y/ANORM
DICOS(3,3,I)=E3Z/ANORM
C
C.... ESTABLISH LOCAL-GLOBAL DIRECTION COSINE MATRIX
C
DICOS(1,2,I)=DICOS(2,3,I)*DICOS(3,1,I)-
1 DICOS(3,3,I)*DICOS(2,1,I)
DICOS(2,2,I)=DICOS(3,3,I)*DICOS(1,1,I)-
1 DICOS(1,3,I)*DICOS(3,1,I)
DICOS(3,2,I)=DICOS(1,3,I)*DICOS(2,1,I)-
1 DICOS(2,3,I)*DICOS(1,1,I)
END IF
END IF
C
C.... WRITE(6,99) I,((DICOS(LR,LC,I),LC=1,3),LR=1,3)
C
C
C.... MASS ASSEMBLY
C.... FOR BEAM ELEMENTS ONLY (TYPE=2) CALCULATE AND ASSEMBLE ELEMENT
C.... MASS
C
IF (IX(10,I).NE.2) RETURN
MTYP=IX(9,I)
RHO=E(1,MTYP)
IF (RHO.EQ.0.) RETURN
RT=RHO*AL(I)*E(7,MTYP)/2.D0
RY=RT*AL(I)*AL(I)/12.D0
RZ=RY
C
C.... RX - MASS MOMENT OF INERTIA ABOUT LOCAL X AXIS IS APPROXIMATE
C....  $RX=RT*(E(5,MTYP)*E(5,MTYP) + E(8,MTYP)*E(8,MTYP))/12.D0$ 
C
RX=RT*2.D0*E(7,MTYP)/12.D0

```

```

C
C.... ASSEMBLE TRANSLATIONAL MASS
C
  DO N=3,4
    KK=IX(N,I)
    NN=(KK-1)*NDGREE
    DO M=1,3
      SMASS(NN+M)=RT+SMASS(NN+M)
    END DO
  END DO
C
C.... ASSEMBLE ROTATIONAL MASS IN GLOBAL COORDINATES
C
  DO M=3,4
    N=IX(M,I)
    DO II=1,3
      DO JJ=1,3
        RMASS(II,JJ,N)=RMASS(II,JJ,N)+DICOS(II,1,I)*DICOS(JJ,1,I)*RX
1          +DICOS(II,2,I)*DICOS(JJ,2,I)*RY+
2          DICOS(II,3,I)*DICOS(JJ,3,I)*RZ
      END DO
    END DO
  END DO
  RETURN
C
  END

```

```

SUBROUTINE BFRGIN (L,ND,XC,YC,ZC,IX,E,UD,FINT,STRS,STRAIN,STRESS,N
IUMEL,IPT,INDEX,SMASS,EUI,EUI,TRAI,TRAJ,EH,AL,IEIGEN,INMESH)
C
Cbrc
Cbrc REPLACED COMMON WITH INCLUDE AND GOTO WITH BLOCK IF AND DO
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.... ROUTINE CALCULATES INTERNAL FORCES FOR THREE-D BEAM
C.... TRAI AND TRAJ CONTAIN I BAR 0 , J BAR 0 AND J HAT 0 FOR NODES I & J
C
INCLUDE 'ADJUST.COM'
COMMON /MATRIX/ NAM,NB,NR
INCLUDE 'NUMINT.COM'
DOUBLE PRECISION FINT(NFINT)

DIMENSION XC(NXC), YC(NYC), IX(14,NIX), UD(NX1), E(12,NE),
1 STRAIN(NSTRAI), SMASS(NSMASS), STRESS(NSTRES),
2 STRS(NSTRS), INDEX(NINDEX), ZC(NZC), EUI(9), EUJ(9),
3 EH(9), TRAI(12), TRAJ(12), AL(NAL), EJBI(3), EBJJ(3),
4 TEMP1(3), TEMP2(3), TEMP3(3), TEMP4(3), TEMP5(3),
5 OMEGI(3,3), OMEGJ(3,3), DISP(2), ROT(6), AFORCE(6),
6 BMOMT(6), IPT(NIPT), INMESH(NINMEH)
N1=IX(1,L)
N2=IX(2,L)
N3=IX(3,L)
N4=IX(4,L)
C
C.... CHECK FOR PRIMARY OR SECONDARY NODES
C
ISKIP=0
JSKIP=0
IF (N1.EQ.N3) ISKIP=1
IF (N2.EQ.N4) JSKIP=1
C
XX=XC(N2)-XC(N1)
YY=YC(N2)-YC(N1)
ZZ=ZC(N2)-ZC(N1)
N1N=(N1-1)*ND
N2N=(N2-1)*ND
N3N=(N3-1)*ND
N4N=(N4-1)*ND
II=N1
JJ=N2

IF (ISKIP.EQ.1) THEN
DIX=UD(N3N+1)
DIY=UD(N3N+2)
DIZ=UD(N3N+3)
ELSE
OMEGI(1,1)=0.D0
OMEGI(1,2)=TRAI(12)
OMEGI(1,3)=TRAI(11)
OMEGI(2,1)=TRAI(12)
OMEGI(2,2)=0.D0
OMEGI(2,3)=TRAI(10)
OMEGI(3,1)=TRAI(11)
OMEGI(3,2)=TRAI(10)
OMEGI(3,3)=0.D0
TEMP1(1)=TRAI(10)
TEMP1(2)=TRAI(11)
TEMP1(3)=TRAI(12)
NAM = 3*3
NB = 3*1
NR = 3*1
CALL GMPRD (EUI,TEMP1,TEMP3,3,3,1)
DIX=UD(N3N+1)+TEMP3(1)-TRAI(7)
DIY=UD(N3N+2)+TEMP3(2)-TRAI(8)
DIZ=UD(N3N+3)+TEMP3(3)-TRAI(9)
UD(N1N+1)=DIX
UD(N1N+2)=DIY
UD(N1N+3)=DIZ
END IF

IF (JSKIP.EQ.1) THEN
DIX=UD(N4N+1)

```

```

    DJY=UD(N4N+2)
    DJZ=UD(N4N+3)
ELSE
    OMEGJ(1,1)=0.D0
    OMEGJ(1,2)=TRAJ(12)
    OMEGJ(1,3)=-TRAJ(11)
    OMEGJ(2,1)=-TRAJ(12)
    OMEGJ(2,2)=0.D0
    OMEGJ(2,3)=TRAJ(10)
    OMEGJ(3,1)=TRAJ(11)
    OMEGJ(3,2)=-TRAJ(10)
    OMEGJ(3,3)=0.D0
    TEMP2(1)=TRAJ(10)
    TEMP2(2)=TRAJ(11)
    TEMP2(3)=TRAJ(12)
    CALL GMPRD (EUJ,TEMP2,TEMP4,3,3,1)
    DJX=UD(N4N+1)+TEMP4(1)-TRAJ(7)
    DJY=UD(N4N+2)+TEMP4(2)-TRAJ(8)
    DJZ=UD(N4N+3)+TEMP4(3)-TRAJ(9)
    UD(N2N+1)=DJX
    UD(N2N+2)=DJY
    UD(N2N+3)=DJZ
END IF

IF (IEIGEN.EQ.1) RETURN
C
C.... FIND RIDID BODY ROTATION AFTER DEFORMATION W/T ORIGINAL COORDINAT
C.... FIND STRAINS AT NODES, HT IS THICKNESS OF BEAM ELEMENT
C
    DIJX=DJX-DIX
    DIJY=DJY-DIY
    DIJZ=DJZ-DIZ
    DX=XX+DIJX
    DY=YY+DIJY
    DZ=ZZ+DIJZ
    AL2=AL(L)*AL(L)
C
C.... FAC = CENTROIDAL AXIS STRAIN
C
    FAC=2.D0*(XX*DIJX+YY*DIJY+ZZ*DIJZ)+DIJX*DIJX+DIJY*DIJY+DIJZ*DIJZ
    ALN2=AL2+FAC
    ALN=DSQRT(ALN2)
    IFLAG=0

    IF (IFLAG.NE.0) THEN
        WRITE (6,901) L,AL(L),ALN
        WRITE (6,902) DIX,DIY,DIZ,DJX,DJY,DJZ
        WRITE (6,903) XX,YY,ZZ,DX,DY,DZ
    END IF
C
C.... EH IS NEW POSITION OF LOCAL AXIS
C
    EH(1)=DX/ALN
    EH(2)=DY/ALN
    EH(3)=DZ/ALN

    DO I=1,3
        I4=3+I
        I3=6+I
        EJBI(I)=EUI(I)*TRAI(4)+EUI(I4)*TRAI(5)+EUI(I3)*TRAI(6)
        EJB(I)=EUJ(I)*TRAJ(4)+EUJ(I4)*TRAJ(5)+EUJ(I3)*TRAJ(6)
    END DO

    DO I=1,3
        I4=3+I
        EH(I4)=(EJBI(I)+EJB(I))/2.D0
    END DO

    EH(7)=EH(2)*EH(6)-EH(3)*EH(5)
    EH(8)=EH(3)*EH(4)-EH(1)*EH(6)
    EH(9)=EH(1)*EH(5)-EH(2)*EH(4)
    ENORM=DSQRT(EH(7)*EH(7)+EH(8)*EH(8)+EH(9)*EH(9))
    EH(7)=EH(7)/ENORM
    EH(8)=EH(8)/ENORM
    EH(9)=EH(9)/ENORM
    EH(4)=EH(8)*EH(3)-EH(9)*EH(2)

```

```

EH(5)=EH(9)*EH(1)-EH(7)*EH(3)
EH(6)=EH(7)*EH(2)-EH(8)*EH(1)
C
C.... DETERMINE DEFORMATION ROTATIONS OF BEAM
C.... EH CONTAINS MU, EUI & EUJ CONTAIN LAMDA I & LAMDA J
C.... CALCULATE DEFORMATION ROTATIONS LOCAL COORDINATES(PHIY,PHIZ ECT.)
C
ROT(1)=(EJBI(2)*EJBJ(3)-EJBI(3)*EJBJ(2))*EH(1)
ROT(1)=ROT(1)+(EJBI(1)*EJBJ(3)-EJBI(3)*EJBJ(1))*EH(2)
ROT(1)=ROT(1)+(EJBI(1)*EJBJ(2)-EJBI(2)*EJBJ(1))*EH(3)
ROT(3)=-(EH(7)*EUI(1)+EH(8)*EUI(2)+EH(9)*EUI(3))*TRAI(1)
ROT(3)=ROT(3)-(EH(7)*EUI(4)+EH(8)*EUI(5)+EH(9)*EUI(6))*TRAI(2)
ROT(3)=ROT(3)-(EH(7)*EUI(7)+EH(8)*EUI(8)+EH(9)*EUI(9))*TRAI(3)
ROT(5)=(EH(4)*EUI(1)+EH(5)*EUI(2)+EH(6)*EUI(3))*TRAI(1)
ROT(5)=ROT(5)+(EH(4)*EUI(4)+EH(5)*EUI(5)+EH(6)*EUI(6))*TRAI(2)
ROT(5)=ROT(5)+(EH(4)*EUI(7)+EH(5)*EUI(8)+EH(6)*EUI(9))*TRAI(3)
ROT(4)=-(EH(7)*EUJ(1)+EH(8)*EUJ(2)+EH(9)*EUJ(3))*TRAJ(1)
ROT(4)=ROT(4)-(EH(7)*EUJ(4)+EH(8)*EUJ(5)+EH(9)*EUJ(6))*TRAJ(2)
ROT(4)=ROT(4)-(EH(7)*EUJ(7)+EH(8)*EUJ(8)+EH(9)*EUJ(9))*TRAJ(3)
ROT(6)=(EH(4)*EUJ(1)+EH(5)*EUJ(2)+EH(6)*EUJ(3))*TRAJ(1)
ROT(6)=ROT(6)+(EH(4)*EUJ(4)+EH(5)*EUJ(5)+EH(6)*EUJ(6))*TRAJ(2)
ROT(6)=ROT(6)+(EH(4)*EUJ(7)+EH(5)*EUJ(8)+EH(6)*EUJ(9))*TRAJ(3)
C
IF (IFLAG.NE.0) THEN
  WRITE (6,904) L,(ROT(LL),LL=3,6)
  WRITE (6,905) (EH(LL),LL=1,9)
  WRITE (6,906) (EUI(LL),LL=1,9)
  WRITE (6,907) (EUJ(LL),LL=1,9)
  WRITE (6,908) (TRAI(LL),LL=1,3)
  WRITE (6,909) (TRAJ(LL),LL=1,3)
END IF
C
C.... CHECK FOR LARGE ROTATIONS
C
DO KK=3,6
  IF(DABS(ROT(KK)).GE.0.15) ROT(KK)=DASIN(ROT(KK))
END DO
C
C.... DETERMINE LOCAL INTERNAL FORCES
C
STREH=FAC/(AL(L)+ALN)/AL(L)
MTYP=IX(9,L)
DISP(1)=ALN
DISP(2)=STREH
NOPT=IX(10,L)
I1=1
ISEC=IX(11,L)
IF (ISEC.NE.0) I1=IPT(ISEC)
CALL LOCFRC (L,DISP,ROT,AFORCE,BMOMT,E,INDEX,I1,STRAIN,STRESS,
1 STRS,SMAS,IX,AL(L),MTYP,NOPT,NUMEL,ZC,INMESH)
C
C.... ACCUMULATE INTERNAL ELEMENTAL FORCES TO FINT
C.... COMPUTE GLOBAL COMPONENTS OF FINT
C
TEMP1(1)=AFORCE(1)
TEMP1(2)=AFORCE(3)
TEMP1(3)=AFORCE(5)
NAM = 3*3
NB = 3*1
NR = 3*1
CALL GMPRD (EH,TEMP1,TEMP2,3,3,1)
TEMP3(1)=AFORCE(2)
TEMP3(2)=AFORCE(4)
TEMP3(3)=AFORCE(6)
CALL GMPRD (EH,TEMP3,TEMP1,3,3,1)

DO I=1,3
  FINT(N3N+I)=FINT(N3N+I)+TEMP2(I)
  FINT(N4N+I)=FINT(N4N+I)+TEMP1(I)
END DO

IF (ISKIP.EQ.1) THEN
  DO I=1,3
    TEMP2(I)=0.D0
  END DO
ELSE

```

```

CALL GTPRD (EUI,TEMP2,TEMP3,3,3,1)
CALL GTPRD (OMEGI,TEMP3,TEMP2,3,3,1)
END IF

```

```

IF (JSKIP.EQ.1) THEN

```

```

  DO I=1,3
    TEMP1(I)=0.D0
  END DO

```

```

ELSE

```

```

  CALL GTPRD (EUJ,TEMP1,TEMP3,3,3,1)
  CALL GTPRD (OMEGJ,TEMP3,TEMP1,3,3,1)
END IF

```

```

TEMP4(1)=BMOMT(1)

```

```

TEMP4(2)=BMOMT(3)

```

```

TEMP4(3)=BMOMT(5)

```

```

TEMP5(1)=BMOMT(2)

```

```

TEMP5(2)=BMOMT(4)

```

```

TEMP5(3)=BMOMT(6)

```

```

CALL GMPRD (EH,TEMP4,TEMP3,3,3,1)

```

```

CALL GTPRD (EUI,TEMP3,TEMP4,3,3,1)

```

```

CALL GMPRD (EH,TEMP5,TEMP3,3,3,1)

```

```

CALL GTPRD (EUJ,TEMP3,TEMP5,3,3,1)

```

```

I=0

```

```

DO J=4,6

```

```

  I=I+1

```

```

  FINT(N3N+J)=FINT(N3N+J)+TEMP4(I)+TEMP2(I)

```

```

  FINT(N4N+J)=FINT(N4N+J)+TEMP5(I)+TEMP1(I)

```

```

END DO

```

```

RETURN

```

C

```

901 FORMAT (2X,5HEL NO,I5,2X,7HL0 + LN,1P2D20.4)

```

```

902 FORMAT (2X,7HDI + DJ/,5X,1P6D20.4)

```

```

903 FORMAT (3X,7HXX + DX/,5X,1P6D20.4)

```

```

904 FORMAT (5X,3HEL=,I5/,5X,3HROT,1P4D20.4)

```

```

905 FORMAT (5X,2HEH/,3(5X,1P3D20.4/))

```

```

906 FORMAT (5X,3HEUL/,3(5X,1P3D20.4/))

```

```

907 FORMAT (5X,3HEUJ/,3(5X,1P3D20.4/))

```

```

908 FORMAT (5X,4HTRAI,1P3D20.4)

```

```

909 FORMAT (5X,4HTRAJ,1P3D20.4)

```

```

END

```

```

SUBROUTINE CROSS (A,B,C,BETA,CMAG,IS)
C
Cbrc
Cbrc CLEANED UP SOME CODE - DJP
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
DIMENSION A(3), B(3), C(3)
C
C.... PROGRAM FORMS CROSS PRODUCT
C.... AND ANGLE BETA. C NORMED TO CMAG
C.... C = A X B
C
C(1)=A(2)*B(3)-A(3)*B(2)
C(2)=A(3)*B(1)-A(1)*B(3)
C(3)=A(1)*B(2)-A(2)*B(1)
AMAG=0.D0
BMAG=0.D0
CMAG=0.D0

DO I=1,3
  AMAG=AMAG+A(I)*A(I)
  BMAG=BMAG+B(I)*B(I)
  CMAG=CMAG+C(I)*C(I)
END DO

AMAG=DSQRT(AMAG)
BMAG=DSQRT(BMAG)
CMAG=DSQRT(CMAG)
BETA=CMAG/(AMAG*BMAG)
IF (DABS(BETA).GT.1.) RETURN
BETA=DASIN(BETA)
IF (IS.EQ.1) RETURN

DO I=1,3
  C(I)=C(I)/CMAG
END DO

RETURN
END

```



```

      SUBROUTINE DECOD (NUM,JJ,NBASE,NTERMS)
C
Cbrc
Cbrc  CLEANED UP SOME CODE - DJP
Cbrc
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION JJ(NTERMS)
C
C.... NUM = NUMBER TO BE DECODED
C.... JJ = RETURNS DECODED TERMS IN NUM 1 TO NTERMS
C.... NBASE = BASE WHICH NUM IS CODED IN
C.... JJ(NTERMS) = NODE NUMBER
C
      ND=NUM
      N=NTERMS-1
      J=NBASE**N
      JJ(NTERMS)=ND/J
      ND=ND-JJ(NTERMS)*J

      DO K=1,N
        J=NBASE**(N-K)
        JJ(K)=ND/J
        ND=ND-JJ(K)*J
      END DO

      RETURN
      END

```

```

SUBROUTINE EIGEN (A,R,N,MV)
C
Cbrc
Cbrc CLEANED UP SOME CODE - DJP
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
COMMON /NEIGEN/ NA,NR
DIMENSION A(NA), R(NR)
RANGE=1.0D-7

IF ((MV-1).NE.0) THEN
  IQ=-N

  DO J=1,N
    IQ=IQ+N
    DO I=1,N
      IJ=IQ+I
      R(IJ)=0.D0
      IF ((I-J).NE.0) CYCLE
      R(IJ)=1.D0
    END DO
  END DO
END IF

ANORM=0.D0

DO I=1,N
  DO J=1,N
    IF ((I-J).EQ.0) CYCLE
    IA=I+(J-J)/2
    ANORM=ANORM+A(IA)*A(IA)
  END DO
END DO

IF (ANORM.GT.0) THEN
  ANORM=DSQRT(2.D0)*DSQRT(ANORM)
  ANRMX=ANORM*RANGE/FLOAT(N)
  IND=0
  THR=ANORM
9  THR=THR/FLOAT(N)
10 L=1
11 M=L+1
12 MQ=(M*M-M)/2
  LQ=(L*L-L)/2
  LM=L+MQ
  IF (DABS(A(LM)).GE.THR) THEN
    IND=1
    LL=L+LQ
    MM=M+MQ
    X=.5D0*(A(LL)-A(MM))
    Y=-A(LM)/DSQRT(A(LM)*A(LM)+X*X)
    IF (X.LT.0) Y=-Y
    SINX=Y/DSQRT(2.D0*(1.D0+(DSQRT(1.D0-Y*Y))))
    SINX2=SINX*SINX
    COSX=DSQRT(1.D0-SINX2)
    COSX2=COSX*COSX
    SINCS=SINX*COSX
    ILQ=N*(L-1)
    IMQ=N*(M-1)

    DO I=1,N
      IQ=(I*I-I)/2
      IF (I.NE.L) THEN

        IF (I.LT.M) THEN
          IM=I+MQ
        ELSE IF (I.GT.M) THEN
          IM=M+IQ
        ELSE
          IF ((MV-1).EQ.0) CYCLE
          ILR=ILQ+I
          IMR=IMQ+I
          X=R(ILR)*COSX-R(IMR)*SINX
          R(IMR)=R(ILR)*SINX+R(IMR)*COSX
          R(ILR)=X

```

```

END IF

IF (I.L.T.L) THEN
  IL=I+LQ
ELSE
  IL=L+IQ
END IF

X=A(IL)*COSX-A(IM)*SINX
A(IM)=A(IL)*SINX+A(IM)*COSX
A(IL)=X
END IF

IF ((MV-1).EQ.0) CYCLE
ILR=ILQ+I
IMR=IMQ+I
X=R(ILR)*COSX-R(IMR)*SINX
R(IMR)=R(ILR)*SINX+R(IMR)*COSX
R(ILR)=X
END DO

X=2.D0*A(LM)*SINCS
Y=A(LL)*COSX2+A(MM)*SINX2-X
X=A(LL)*SINX2+A(MM)*COSX2+X
A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)
A(LL)=Y
A(MM)=X
END IF

IF (M.NE.N) THEN
  M=M+1
  GO TO 12
END IF

IF (L.NE.(N-1)) THEN
  L=L+1
  GO TO 11
END IF

IF (IND.EQ.1) THEN
  IND=0
  GO TO 10
END IF

IF (THR.GT.ANRMX) GO TO 9
END IF

IQ=-N

DO I=1,N
  IQ=IQ+N
  LL=I+(I*I-I)/2
  JQ=N*(I-2)

  DO J=1,N
    JQ=JQ+N
    MM=J+(J*J-J)/2
    IF (A(LL).GE.A(MM)) CYCLE
    X=A(LL)
    A(LL)=A(MM)
    A(MM)=X
    IF (MV.EQ.1) CYCLE
  DO K=1,N
    ILR=IQ+K
    IMR=JQ+K
    X=R(ILR)
    R(ILR)=R(IMR)
    R(IMR)=X
  END DO
END DO
END DO

RETURN
END
SUBROUTINE ELOUT (STRS,IX,INDEX,NUMEL,IPTS)

```

```

C
Cbrc
Cbrc CLEANED UP CODE - DJP
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
INCLUDE 'ADJUST.COM'
DIMENSION STRS(NSTRS), IX(14,NIX), INDEX(NINDEX), IPTS(NIPT)

WRITE (6,901)
CON1=4.448D5
CON2=.394D0
CON3=CON2/CON1
CON4=CON2**3
IFLAG=0
IPLATE=0

DO JE=1,NUMEL
  NOPT=IX(10,JE)
  IND=INDEX(JE)
  IF (NOPT.EQ.1) THEN
    N=IND+14
    WRITE (6,902) JE,STRS(N)
    STRS(N)=STRS(N)/CON1
    WRITE (6,903) STRS(N)
    STRS(N)=STRS(N)*CON1
  ELSE IF (NOPT.EQ.2 .OR. NOPT.EQ.3) THEN
    IF (NOPT.EQ.2) THEN
      N=IND+26
      IF (IX(11,JE).NE.0) IFLAG=IX(11,JE)
    ELSE IF (NOPT.EQ.3) THEN
      N=IND+24
    END IF
    NEND=N+7
    WRITE (6,904) JE,(STRS(K),K=N,NEND)
    DO K=1,3
      I=K-1
      STRS(N+I)=STRS(N+I)/CON1
    END DO
    DO K=4,8
      I=K-1
      STRS(N+I)=STRS(N+I)*CON3
    END DO
    WRITE (6,905) (STRS(K),K=N,NEND)
    DO K=1,3
      I=K-1
      STRS(N+I)=STRS(N+I)*CON1
    END DO
    DO K=4,8
      I=K-1
      STRS(N+I)=STRS(N+I)/CON3
    END DO
  ELSE IF (NOPT.EQ.4) THEN
    N=IND+36
    WRITE (6,906) JE,STRS(N),STRS(N+1),STRS(N+3)
    STRS(N)=STRS(N)/(CON1*CON2*CON2)
    STRS(N+1)=STRS(N+1)*CON4
    STRS(N+3)=STRS(N+3)/CON1
    WRITE (6,907) STRS(N),STRS(N+1),STRS(N+3)
    STRS(N)=STRS(N)*CON1*CON2*CON2
    STRS(N+1)=STRS(N+1)/CON4
    STRS(N+3)=STRS(N+3)*CON1
  ELSE IF (NOPT.EQ.5) THEN
    IPLATE=1
  END IF
END DO

IF (IFLAG.NE.0) THEN
  WRITE (6,908)
  DO I=1,NUMEL
    IF (IX(10,I).EQ.5) CYCLE
    ISEC=IX(11,I)
    I1=1
    IF (ISEC.NE.0) I1=IPTS(ISEC)
    WRITE (6,909) I
    IF (I1.EQ.1) THEN

```

```

      IND=INDEX(I)+24
      WRITE (6,911) STRS(IND),STRS(IND+1)
      CYCLE
    END IF
    IND=INDEX(I)+41
    DO K=1,2
      WRITE (6,910) K
      DO M=1,I1
        N1=IND+M-1+(K-1)*I1
        N2=2*I1+N1
        N3=4*I1+N1
        WRITE (6,911) STRS(N1),STRS(N2),STRS(N3)
      END DO
    END DO
  END DO
  END IF

  IF (IPLATE.EQ.0) RETURN
  WRITE (6,912)

  DO I=1,NUMEL
    IP=IX(11,I)
    IND=INDEX(I)-1
    WRITE (6,913) I
    DO KP=1,IP
      WRITE (6,914) KP
      DO IS=1,3
        NN=IS+3
        N1=IND+30+3*(7*(KP-1)+IS-1)+1
        N2=N1+9
        N3=N2+9-2*(IS-1)
        WRITE (6,915) NN,STRS(N1),STRS(N1+1),STRS(N1+2),STRS(N2),
1          STRS(N2+1),STRS(N2+2),STRS(N3)
      END DO
    END DO
  END DO

  RETURN
C
901 FORMAT (10X,20HLOCAL ELEMENT FORCES,/,5X,11HELEMENT NO.,6X,7H(UN
   IITS),7X,11HAXIAL FORCE,10X,9HY - SHEAR,10X,9HZ - SHEAR,10X,10HX -
   2MOMENT,/,36X,11HYI - MOMENT,10X,11HYJ - MOMENT,8X,11HZI - MOMENT,8
   3X,11HZJ - MOMENT,/)
902 FORMAT (5X,I5,11X,9H(DYNE-CM),G16.6,/)
903 FORMAT (21X,8H(LBF-IN),1X,G16.6,/)
904 FORMAT (5X,I5,11X,9H(DYNE-CM),4(G16.6,5X),/,30X,4(G16.6,5X),/)
905 FORMAT (21X,8H(LBF-IN),1X,4(G16.6,5X),/,30X,4(G16.6,5X),/)
906 FORMAT (5X,I5,11X,9H(DYNE-CM),5X,10HPRESSURE =,1PD12.4,5X,8HVOLUME
   1 =,1PD12.4,5X,7HFORCE 1,1PD12.4,/)
907 FORMAT (21X,8H(LBF-IN),6X,10HPRESSURE =,1PD12.4,5X,8HVOLUME =,1PD1
   12.4,5X,7HFORCE =,1PD12.4,/)
908 FORMAT (1H1,6X,11HELEMENT NO.,18X,7HSECTION,6X,6HSTRAIN,19X,6HSTRE
   1SS,16X,12HYIELD STRESS)
909 FORMAT (12X,I5)
910 FORMAT (35X,I5)
911 FORMAT (36X,3(5X,1PD20.4))
912 FORMAT (1H1,9X,33HSTRAINS,STRESSES AND YIELD VALUES//1X,7HELEMENT/
   16X,5HLAYER,9X,5HEPS-X,8X,5HEPS-Y,8X,6HEPS-XY,7X,5HSTR-X,8X,5HSTR-Y
   2,8X,6HSTR-XY,6X,5HYIELD/,10X,5HPOINT)
913 FORMAT (15,I4)
914 FORMAT (5X,I4)
915 FORMAT (10X,I3,2X,1PD13.5)
  END

```

```

SUBROUTINE FRCIN (NUMEL,NDGREE,XC,YC,ZC,IX,E,UD,FINT,STRS,ST
IRAIN,STRESS,IPTS,INDEX,BLAMB,SMASS,DICOS,NPRI,AL,IEIGEN,INMESH)
Cbrc
Cbrc CLEANED UP CODE - DJP
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)

DOUBLE PRECISION FINT(NFINT)

INCLUDE 'ADJUST.COM'
DIMENSION XC(NXC), YC(NYC), ZC(NZC), IX(14,NIX), E(12,NE), UD(
1  NX1), STRS(NSTRS), DICOS(9,NDICOS), STRAIN(NSTRAI),
2  STRESS(NSTRES), INDEX(NINDEX), BLAMB(NNBLAM), SMASS(
3  NSMASS), AL(NAL), IPTS(NIPT), INMESH(NINMEH)
C
C.... THIS SUBROUTINE CALCULATES THE INTERNAL FORCES IN ALL OF THE
C.... ELEMENTS AND RETURNS THEM IN FINT
C
IF (IEIGEN.NE.1) THEN
  NEQ=NPRI*NDGREE
  ISTIF=0
C
C.... ZERO OUT FINT ARRAY
C
DO LEN=1,NEQ
  FINT(LEN)=0.D0
END DO
END IF
C
C.... SET UP LOCAL ARRAY ENTRIES FOR BFRCIN AND SFRCIN
C
C
DO JE=1,NUMEL
  NOPT=IX(10,JE)
  NP1=3
  IF (NOPT.EQ.4) NP1=7
  IF (NOPT.EQ.5) NP1=4
  I=(IX(NP1,JE)-1)*9+1
  J=(IX(NP1+1,JE)-1)*9+1
  K=(IX(NP1+2,JE)-1)*9+1
  M=INDEX(JE)
  MB=M+12
  MS=M+6
  MPV=M+18
  MPL2=M+9
  MPL3=MPL2+9
  MPSIDE=M+27
  IF (NOPT.EQ.1) THEN
    CALL SFRCIN (JE,NDGREE,XC,YC,ZC,IX,E,UD,FINT,STRS,STRAIN,
1    STRESS,NUMEL,INDEX,SMASS,BLAMB(I),BLAMB(J),
2    STRS(M),STRS(MS),DICOS(1,JE),AL,IEIGEN,
3    INMESH)
  ELSE IF (NOPT.EQ.2 .OR. NOPT.EQ.3) THEN
    CALL BFRCIN (JE,NDGREE,XC,YC,ZC,IX,E,UD,FINT,STRS,STRAIN,
1    STRESS,NUMEL,IPTS,INDEX,SMASS,BLAMB(I),BLAMB(J),
2    STRS(M),STRS(MB),DICOS(1,JE),AL,IEIGEN,INMESH)
  ELSE
    CYCLE
  END IF
END DO

RETURN
END

```

```

SUBROUTINE FREEFD(KONWAV,NUMDIS,NODDIS,NNODE,NDGREE,XC,YC,ZC,
1      UD,UD1,UD2,FORCD,INMESH,BETA )

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

INCLUDE 'ADJUST.COM'
INCLUDE 'DYNAM.COM'
DIMENSION UD(1),UD1(1),UD2(1),FORCD(1),NODDIS(1),INMESH(1)
C
C
C FREEFD FOR HYBRID II PENDULUM TEST. INITIAL VELOCITY = 19.5 ft/sec AT WAND
C PRESCRIBED ACCELERATION PROFILE CONSISTS OF
C "HALF-SINE" FOR 0.LE.T.LT.T1
C "1-COS+CONSTANT" FOR T1.LE.T.LT.T2
C "A CONSTANT" FOR T2.LE.T
C
C V0 IS THE INITIAL X-VELOCITY OF BASE POINT ==> FLEXION <==
C
Cbrc
Cbrc INITIALIZE VARIABLES IF THE FIRST TIME STEP
Cbrc
IF (TIME .LT. DELT) THEN
  G = 980.66352D0
  VO = 658.1
  NCGTX = 6*(INMESH(4)-1)+1
  NPNTX = 6*(INMESH(1)-1)+1
  UD1(NCGTX) = 710.5
  CALL ICIF(TIME,DUMMY,2)
  RETURN
END IF

CALL ICIF(TIME,ACC,0)
CALL ICIF(TIME,VEL,1)
CALL ICIF(TIME,DIS,2)
UD (NPNTX) = DIS*G + VO*TIME
UD1(NPNTX) = VEL*G + VO
UD2(NPNTX) = ACC*G

RETURN
END

```

```

SUBROUTINE GMPRD (A,B,R,N,M,L)
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.....
C..... SUBROUTINE GMPRD
C..... PURPOSE
C..... MULTIPLY TWO GENERAL MATRICES TO FORM A RESULTANT GENERAL
C..... MATRIX
C..... USAGE
C..... CALL GMPRD(A,B,R,N,M,L)
C..... DESCRIPTION OF PARAMETERS
C..... A - NAME OF FIRST INPUT MATRIX
C..... B - NAME OF SECOND INPUT MATRIX
C..... R - NAME OF OUTPUT MATRIX
C..... N - NUMBER OF ROWS IN A
C..... M - NUMBER OF COLUMNS IN A AND ROWS IN B
C..... L - NUMBER OF COLUMNS IN B
C..... REMARKS
C..... ALL MATRICES MUST BE STORED AS GENERAL MATRICES
C..... MATRIX R CANNOT BE IN THE SAME LOCATION AS MATRIX A
C..... MATRIX R CANNOT BE IN THE SAME LOCATION AS MATRIX B
C..... NUMBER OF COLUMNS OF MATRIX A MUST BE EQUAL TO NUMBER OF ROW
C..... OF MATRIX B
C..... SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C..... NONE
C..... METHOD
C..... THE M BY L MATRIX B IS PREMULTIPLIED BY THE N BY M MATRIX A
C..... AND THE RESULT IS STORED IN THE N BY L MATRIX R.
C.....
C
COMMON /MATRIX/ NA,NB,NR
DIMENSION A(NA), B(NB), R(NR)
C
IR=0
IK=-M
DO 1 K=1,L
IK=IK+M
DO 1 J=1,N
IR=IR+1
JI=J-N
IB=IK
R(IR)=0
DO 1 I=1,M
JI=JI+N
IB=IB+1
1 R(IR)=R(IR)+A(JI)*B(IB)
RETURN
END

```



```

SUBROUTINE GTPRD (A,B,R,N,M,L)
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.....
C..... SUBROUTINE GTPRD
C..... PURPOSE
C.....   PREMULTIPLY A GENERAL MATRIX BY THE TRANSPOSE OF ANOTHER
C.....   GENERAL MATRIX
C..... USAGE
C.....   CALL GTPRD(A,B,R,N,M,L)
C..... DESCRIPTION OF PARAMETERS
C.....   A - NAME OF FIRST INPUT MATRIX
C.....   B - NAME OF SECOND INPUT MATRIX
C.....   R - NAME OF OUTPUT MATRIX
C.....   N - NUMBER OF ROWS IN A AND B
C.....   M - NUMBER OF COLUMNS IN A AND ROWS IN R
C.....   L - NUMBER OF COLUMNS IN B AND R
C..... REMARKS
C.....   MATRIX R CANNOT BE IN THE SAME LOCATION AS MATRIX A
C.....   MATRIX R CANNOT BE IN THE SAME LOCATION AS MATRIX B
C.....   ALL MATRICES MUST BE STORED AS GENERAL MATRICES
C..... SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C.....   NONE
C..... METHOD
C.....   MATRIX TRANSPOSE OF A IS NOT ACTUALLY CALCULATED. INSTEAD,
C.....   ELEMENTS OF MATRIX A ARE TAKEN COLUMNWISE RATHER THAN
C.....   ROWWISE FOR POSTMULTIPLICATION BY MATRIX B.
C.....
C
COMMON /MATRIX/ NA,NB,NR
DIMENSION A(NA), B(NB), R(NR)
C
IR=0
IK=-N
DO 1 K=1,L
  IJ=0
  IK=IK+N
  DO 1 J=1,M
    IB=IK
    IR=IR+1
    R(IR)=0
    DO 1 I=1,N
      IJ=IJ+1
      IB=IB+1
1    R(IR)=R(IR)+A(IJ)*B(IB)
  RETURN
END

```

```

SUBROUTINE ICIF (TIME,VALUE,NI)
Cbrc
Cbrc CODE CLEAN-UP - DJP
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)

DIMENSION T(26),F(26),FP(26),F1(25),F2(25)
NJ=NI+1
C
C.... READ IN DATA AND INITIALIZE INTEGRATION ROUTINE
C

IF (TIME .LE. 0.D0) THEN
  READ (5,901) NPTS,F1(1),F2(1)
  READ (5,902) (T(K),F(K),FP(K),K=1,NPTS)
  WRITE (6,903) NPTS,F1(1),F2(1)
  WRITE (6,904) (T(K),F(K),FP(K),K=1,NPTS)
  DO K=2,NPTS
    TK=T(K)
    CALL ICIF2 (TK,0.D0,2,NPTS,T,F,FP,F1,F2)
    F1(K)=0.D0
    CALL ICIF2 (TK,0.D0,3,NPTS,T,F,FP,F1,F2)
    F2(K)=0.D0
  END DO
END IF

CALL ICIF2 (TIME,VALUE,NJ,NPTS,T,F,FP,F1,F2)

RETURN

901 FORMAT (I5,5X,2D10.0)
902 FORMAT (3D10.0)
903 FORMAT (////,10X,26HNUMERICAL INTEGRATION DATA,/,5X,18HNO. OF POI
  INTS =,I5,/,5X,18HINITIAL VELOCITY =,1PD15.6,/,5X,18HINITIAL DIS
  2P =,1PD15.6,/,15X,4HTIME,15X,9HFCN-VALUE,11X,14HFCN-DERIVATIVE
  3,/)
904 FORMAT (5X,1P3D20.4,/)

END

```

```

      SUBROUTINE ICIF2 (TIME,VALUE,NJ,NPTS,T,F,FP,F1,F2)
Cbrc
Cbrc CODE CLEAN-UP - DJP
Cbrc
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      DIMENSION T(26),F(26),FP(26),F1(25),F2(25),G(4),PHI(4)
C
C.... DETERMINE INTERVAL NUMBER IN
C
      IN=1
      DO K=2,NPTS
        IF (TIME .LE. T(K)) EXIT
        IN=IN+1
      END DO
C
C.... CHECK UPPER BOUND
C
      IF (IN.GE.NPTS) THEN
        WRITE (6,901) TIME,T(NPTS)
        IN=NPTS-1
      END IF
C
C.... SET UP FUNCTIONAL VALUES FOR APPROPRIATE INTERVAL
C
      TI=T(IN)
      TJ=T(IN+1)
      DT=TJ-TI

      G(1)=F(IN)
      G(2)=FP(IN)
      G(3)=F(IN+1)
      G(4)=FP(IN+1)
      X=TIME-TI
C
C.... TRANSFER TO APPROPRIATE NUMBER OF INTEGRATIONS
C
      IF (NJ.EQ.1) THEN
C
C.... FUNCTIONAL VALUE
C
        T1=1.D0
        XL=X/DT
        XL2=XL*XL
        XL3=XL*XL2
        CON=0.D0
      ELSE IF (NJ.EQ.2) THEN
C
C.... ONE INTEGRATION
C
        T1=X
        XL=X**2/(2.D0*DT)
        XL2=X**3/(3.D0*DT**2)
        XL3=X**4/(4.D0*DT**3)
        CON=F1(IN)
      ELSE IF (NJ.EQ.3) THEN
C
C.... TWO INTEGRATIONS
C
        T1=X**2/2.D0
        XL=X**3/(6.D0*DT)
        XL2=X**4/(12.D0*DT**2)
        XL3=X**5/(20.D0*DT**3)
        CON=X*F1(IN)+F2(IN)
      END IF
C
C.... EVALUATION OF APPROPRIATE FUNCTION
C
      PHI(1)=T1-3.D0*XL2+2.D0*XL3
      PHI(2)=(XL-2.D0*XL2+XL3)*DT
      PHI(3)=T1-PHI(1)
      PHI(4)=(XL3-XL2)*DT
      VALUE=CON
      DO K=1,4
        VALUE=VALUE+PHI(K)*G(K)
      END DO

```

RETURN

C

901 FORMAT (///,10X,34H WARNING - AN INTEGRATION POINT X=,D20.8/,  
128X,35HIS GREATER THAN THE MAXIMUM VALUE =,D20.8/,28X,69HIT IS  
2 ASSUMED THIS X IS IN THE LAST INTERVAL EXECUTION CONTINUING,//)

END

```

      SUBROUTINE INCODE (NUM,JJ,NBASE,NTERMS)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      Cbrc CODE CLEAN-UP - DJP
      Cbrc
      DIMENSION JJ(NTERMS)
C
      C.... NUM = RETURNS CODED NUMBER
      C.... JJ = TERMS TO BE CODED INTO NUM 1 TO NTERMS
      C.... NBASE = BASE USED TO CODE NUM
C
      NUM=0
      DO K=1,NTERMS
        NUM=NUM+JJ(K)*NBASE**(NTERMS-K)
      END DO
      RETURN
      END

```

```

SUBROUTINE IODECLS(INP, OUT, PLT)
INTEGER INP, OUT, PLT, LUTRM, LUIN, LUOUTP, LUPLT
CHARACTER*1 ANS, QUOTE
CHARACTER*64 NAINP, NAOUT, NAPLT
CHARACTER*80 ITITLE
LOGICAL IEXIST
COMMON /DEVS/LUTRM, LUIN, LUOUTP, LUPLT
COMMON /LABELS/NAINP, NAOUT, NAPLT
QUOTE = ""
LUTRM = 0
LUIN = 5
LUOUTP = 6
LUPLT = 7
C
C  TERMINAL: LOGICAL UNIT 0
C  INPUT: LOGICAL UNIT 1 (DATA FILE)
C  OUTPUT: LOGICAL UNIT 2
C  PLOT FILE: LOGICAL UNIT 3
C
IEXIST = .FALSE.
IF (INP .NE. 0) THEN
  WRITE(LUTRM,(/,5X,"ENTER INPUT FILE NAME.))
5  READ(LUTRM,(A64)) NAINP
  INQUIRE(FILE=NAINP,EXIST=IEXIST)
  IF (IEXIST) THEN
    OPEN(LUIN,FILE=NAINP,STATUS='OLD')
    WRITE(LUTRM,(/,5X,"INPUT FILE NAME: ",A64/,5X,
*   "NOT FOUND")) NAINP
    WRITE(LUTRM,(/,5X,"ENTER A NEW FILE NAME.))
    GOTO 5
  END IF
END IF
IEXIST = .FALSE.
IF (OUT .NE. 0) THEN
  WRITE(LUTRM,(/,5X,"ENTER OUTPUT FILE NAME.))
10  READ(LUTRM,(A64)) NAOUT
  INQUIRE(FILE=NAOUT,EXIST=IEXIST)
  IF (IEXIST) THEN
    WRITE(LUTRM,(5X,"*** FILE NAME: ",A64/,5X,
*   "ALREADY EXISTS. DO YOU WISH TO OVERWRITE IT? (Y/N)"))
    NAOUT
    READ(LUTRM,(A1)) ANS
    IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') THEN
      OPEN(LUOUTP,FILE=NAOUT,IOSTAT=IERR,STATUS='OLD',
*   BLOCKSIZE=2048)
      REWIND(UNIT=LUOUTP)
    ELSE
      WRITE(LUTRM,(/,5X,"ENTER A NEW FILE NAME"))
      GOTO 10
    END IF
  ELSE
    OPEN(LUOUTP,FILE=NAOUT,IOSTAT=IERR,STATUS='NEW',
*   BLOCKSIZE=2048)
  END IF
  WRITE(LUTRM,(/,5X,"OUTPUT FILE NAME: ",A64)) NAOUT
  WRITE(LUTRM,(/,5X,"DO YOU WISH TO WRITE A TITLE LINE ON THE",
*   " OUTPUT FILE? (Y/N)"))
  READ(LUTRM,(A1)) ANS
  IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') THEN
    WRITE(LUTRM,(/,5X,"ENTER A TITLE LINE.))
    READ(LUTRM,(A80)) ITITLE
    WRITE(LUOUTP,(A1,A80,A1)) QUOTE,ITITLE,QUOTE
  END IF
END IF
IEXIST = .FALSE.
IF (PLT .NE. 0) THEN
  WRITE(LUTRM,(/,5X,"ENTER OUTPUT PLOT FILE NAME.))
15  READ(LUTRM,(A64)) NAPLT
  INQUIRE(FILE=NAPLT,EXIST=IEXIST)
  IF (IEXIST) THEN
    WRITE(LUTRM,(5X,"*** FILE NAME: ",A64/,5X,
*   "ALREADY EXISTS. DO YOU WISH TO OVERWRITE IT? (Y/N)"))
    NAPLT
    READ(LUTRM,(A1)) ANS

```

```

IF(ANS.EQ.'Y'.OR.ANS.EQ.'y') THEN
  OPEN(LUPLT,FILE=NAPLT,IOSTAT=IERR,STATUS='OLD')
  REWIND(UNIT=LUPLT)
ELSE
  WRITE(LUTRM,'(//,5X,"ENTER A NEW FILE NAME")')
  GOTO 15
END IF
ELSE
  OPEN(LUPLT,FILE=NAPLT,IOSTAT=IERR,STATUS='NEW')
END IF
  WRITE(LUTRM,'(//,5X,"PLOT FILE NAME: ",A64)) NAPLT
END IF
RETURN
END

```

```

SUBROUTINE LOCFRC (JE,DISP,ROT,AFORCE,BMOMT,E,INDEX,I1,STRAIN,STRE
ISS,STRS,SMASS,IX,EL,MTYP,NOPT,NUMEL,ZC,INMESH)
C
Cbrc
Cbrc ADDED INCLUDES AND REMOVED GO TO'S - DJP
Cbrc
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)

  INCLUDE 'ADJUST.COM'
  INCLUDE 'DYNAM.COM'
  INCLUDE 'CONTRL.COM'
C
  DIMENSION DISP(2), ROT(6), AFORCE(6), BMOMT(6), E(12,NE), INDEX(
1    NINDEX), STRAIN(NSTRAI), STRESS(NSTRES), SMASS(
2    NSMASS), IX(14,NIX), STRS(NSTRS)
  DIMENSION ZC(NZC),INMESH(NINMEH)
C
C
  DATA ISKIP/0/
C
C.... NOPT=1 - 3-D AXIAL SPRING ELEMENT
C.... NOPT=2 - 3-D LINEAR, ELASTIC RECTANGULAR BEAM ELEMENT
C.... NOPT=3 - 3-D SPINAL DISK BEAM ELEMENT.
C
C MODIFIED FEB, 1978
C A. CUBIC STIFFENING OF SPRING ELEMENTS
C   1. EXPLICIT ONLY
C   2. STIFFNESS PROPORTIONAL DAMPING ONLY
C
C B. CALL TO INJURY CRITERIA SUBROUTINE
C   1. IF KONTRL(15) = 1
C
  IF (NOPT.EQ.1) IND=INDEX(JE)+12
  IF (NOPT.EQ.2) IND=INDEX(JE)+24
  IF (NOPT.EQ.3) IND=INDEX(JE)+34
  INDB=INDEX(JE)+36
  SDPY=E(9,MTYP)
  SDPZ=E(10,MTYP)
  EPSOLD=STRS(IND)
  STRS(IND)=DISP(2)
  IF (I1.GT.1) GO TO 16
C
  IF (NOPT.LE.2) THEN
    STRAIN(1)=DISP(2)
    STRAIN(2)=DISP(2)
C
C.... CALCULATE ELEMENT STRESSES
C.... CALL STRES(JE,MTYP,STRAIN,STRESS,E,STRS,I1,IND)
C
    STRESS(1)=E(2,MTYP)*STRAIN(1)
C
C.... STORE STRESSES AND STRAINS IN STRS
C
    STRS(IND)=STRAIN(1)
    STRS(IND+1)=STRESS(1)
  END IF
C
C.... CALCULATE ELEMENT STIFFNESSES
C
  CSTIF=0.D0

  IF (NOPT.EQ.1) THEN
    ASTIF=E(2,MTYP)*E(7,MTYP)/EL
    IF (E(7,MTYP).LT.0.) ASTIF=E(2,MTYP)
    CSTIF=0.D0
    CSTIF=E(3,MTYP)
  ELSE IF (NOPT.EQ.2) THEN
    ASTIF=E(2,MTYP)*E(7,MTYP)/EL
    IF (E(7,MTYP).LT.0.) ASTIF=E(2,MTYP)
    CSTIF=0.D0
    YM=E(2,MTYP)
    POIS=E(6,MTYP)
    H=E(8,MTYP)
    T=E(5,MTYP)
    SM=YM/(2.D0*(1.D0+POIS))
    BYSTIF=YM*(H*T**3)/(EL*12.D0)

```



```

BZSTIF=YM*(T*H**3)/(EL*12.D0)
IF (H.GT.T) THEN
  TSTIF=.237D0*SM*(H*T**3)/EL
ELSE
  TSTIF=.237D0*SM*(T*H**3)/EL
END IF
ELSE IF (NOPT.EQ.3) THEN
  ASTIF=E(1,MTYP)
  TSTIF=E(2,MTYP)
  AYROT=(DABS(ROT(3))+DABS(ROT(4)))/2.D0
  AZROT=(DABS(ROT(5))+DABS(ROT(6)))/2.D0
  BYSTIF=E(3,MTYP)+E(5,MTYP)*AYROT*AYROT
  BZSTIF=E(4,MTYP)+E(6,MTYP)*AZROT*AZROT
END IF
C
C.... CALCULATE INTERNAL ELEMENT DAMPING FORCES
C
  FDAMP=0.D0
  DMIY=0.D0
  DMJY=0.D0
  DMIZ=0.D0
  DMJZ=0.D0
C
C.... PERCENT CRITICAL DAMPING
C
  IF (KONTRL(8).LE.0) THEN
    PCAD=E(11,MTYP)
    N3=IX(3,JE)
    N4=IX(4,JE)
    N3N=6*(N3-1)+1
    N4N=6*(N4-1)+1

    IF (PCAD.NE.0.) THEN
      AMASS=(SMASS(N3N)+SMASS(N4N))/2.D0
      DDOT=(DISP(2)-EPSOLD)*EL/DEL T
      FDAMP=2.D0*PCAD*DDOT*DSQRT(AMASS*ASTIF)
    END IF

    IF (NOPT.NE.1) THEN
      PCBD=E(12,MTYP)
      IF (PCBD.NE.0.) THEN
        AYMSS=(SMASS(N3N+4)+SMASS(N4N+4))/2.D0
        AZMSS=(SMASS(N3N+5)+SMASS(N4N+5))/2.D0
        DYDOT=(ROT(3)-STRS(INDB+1)-ROT(4)+STRS(INDB+2))/DEL T
        DZDOT=(ROT(5)-STRS(INDB+3)-ROT(6)+STRS(INDB+4))/DEL T
        DMIY=2.D0*PCBD*DYDOT*DSQRT(AYMSS*BYSTIF)
        DMJY=-DMIY
        DMIZ=2.D0*PCBD*DZDOT*DSQRT(AZMSS*BZSTIF)
        DMJZ=-DMIZ
      END IF
    END IF
  END IF
C
C.... VISCOUS DAMPING
C
  ELSE IF (KONTRL(8).GT.0) THEN
    PVAD=E(11,MTYP)
    IF (NOPT.EQ.1) THEN
      IF (PVAD.NE.0) FDAMP=PVAD*ASTIF*(DISP(2)-EPSOLD)*EL/DEL T
    ELSE
      IF (PVAD.NE.0) FDAMP=PVAD*ASTIF*(DISP(2)-EPSOLD)*EL/DEL T
    PVBD=E(12,MTYP)
    IF (PVBD.NE.0.) THEN
      DYI=(ROT(3)-STRS(INDB+1))/DEL T
      DYJ=(ROT(4)-STRS(INDB+2))/DEL T
      DZI=(ROT(5)-STRS(INDB+3))/DEL T
      DZJ=(ROT(6)-STRS(INDB+4))/DEL T
      DMIY=PVBD*BYSTIF*((4.D0+SDPZ)*DYI+
1      (2.D0-SDPZ)*DYJ)/(1.D0+SDPZ)
      DMJY=PVBD*BYSTIF*((2.D0-SDPZ)*DYI+
1      (4.D0+SDPZ)*DYJ)/(1.D0+SDPZ)
      DMIZ=PVBD*BZSTIF*((4.D0+SDPY)*DZI+
1      (2.D0-SDPY)*DZJ)/(1.D0+SDPY)
      DMJZ=PVBD*BZSTIF*((2.D0-SDPY)*DZI+
1      (4.D0+SDPY)*DZJ)/(1.D0+SDPY)
    END IF
  END IF

```

```

      END IF
C
C.... CALCULATE ELEMENT INTERNAL FORCES
C
      IF (NOPT.NE.1) THEN
        TORC=0.D0
        DNOL=DISP(2)*EL
C
C.... SPRING ELEMENT TORC IS TENSION OR COMPRESSION OPTION
C.... TORC < 0 SPRING IS COMPRESSION ONLY
C.... TORC = 0 SPRING IS BOTH TENSION AND COMPRESSION
C.... TORC > 0 SPRING IS TENSION ONLY
C
      ELSE IF (NOPT.EQ.1) THEN
        TORC=E(1,MTYP)
        SLACK=E(4,MTYP)
        DNOL=(DISP(2)-SLACK)*EL

        IF (TORC.LT.0) THEN
          IF (DNOL.GE.0.) DNOL=0.D0
        ELSE IF (TORC.GT.0) THEN
          IF (DNOL.LE.0.) DNOL=0.D0
        END IF
      END IF

      AFORCE(2)=(ASTIF+CSTIF*DNOL*DNOL)*DNOL+FDAMP
      IF (TORC.LT.0..AND.AFORCE(2).GT.0.) AFORCE(2)=0.D0
      IF (TORC.GT.0..AND.AFORCE(2).LT.0.) AFORCE(2)=0.D0
      AFORCE(1)=-AFORCE(2)
      IF (NOPT.EQ.1) GO TO 17
C
C.... CALCULATE INTERNAL MOMENTS
C
      BMOMT(2)=TSTIF*ROT(1)
      BMOMT(1)=-BMOMT(2)
      BMOMT(3)=DMY+BYSTIF*((4.D0+SDPZ)*ROT(3)+(2.D0-SDPZ)*ROT(4))/
1      (1.D0+SDPZ)
      BMOMT(4)=DMJY+BYSTIF*((2.D0-SDPZ)*ROT(3)+(4.D0+SDPZ)*ROT(4))/
1      (1.D0+SDPZ)
      BMOMT(5)=DMIZ+BZSTIF*((4.D0+SDPY)*ROT(5)+(2.D0-SDPY)*ROT(6))/
1      (1.D0+SDPY)
      BMOMT(6)=DMJZ+BZSTIF*((2.D0-SDPY)*ROT(5)+(4.D0+SDPY)*ROT(6))/
1      (1.D0+SDPY)
C
      AFORCE(4)=- (BMOMT(5)+BMOMT(6))/EL
      AFORCE(3)=-AFORCE(4)
      AFORCE(6)= (BMOMT(3)+BMOMT(4))/EL
      AFORCE(5)=-AFORCE(6)
C
      IF (KONTRL(15).EQ.0) GO TO 17

      IF (TIME.LT.DELT .AND. ISKIP.LE.0) THEN
        READ(5,100) JSTART,JEND
        WRITE(6,100) JSTART,JEND
        NB=JEND-JSTART
        ISKIP=1
      END IF

      IF (JE.LT.JSTART.OR.JE.GT.JEND) GO TO 17
      CALL SPINIF (JE,JSTART,JEND,NB,AFORCE,BMOMT,ZC,NZC,INMESH,NINMEH)
      GO TO 17
C
C.... ELASTO - PLASTIC BEAMS
C
      16 IND=INDEX(JE)+41
      IGO=0
      ISEC=IX(11,JE)
C
C.... CALCULATE AND STORE INTERNAL ENERGIES
C
      17 IKE=INDEX(NUMEL+1)
      IF (NOPT.NE.1) IND=INDEX(JE)+24
      IADD=11
      IF (NOPT.EQ.1) IADD=3
C
      IABE=2

```

```

IF(NOPT.EQ.3) IABE=0
DLN=DISP(2)*EL
STRS(IKE+MTYP)=STRS(IKE+MTYP)+.5D0*(AFORCE(2)+STRS(IND+IABE))
* *(DLN-STRS(IND+IADD))
STRS(IND+IADD)=DLN

IF (NOPT.NE.1) THEN
  IADD=1
  IF (NOPT.EQ.2) IADD=3
  ROT(2)=ROT(1)
  DO KK=2,6
    KI=IND+10+KK
    DRN=ROT(KK)-STRS(KI)
    STRS(IKE+MTYP)=STRS(IKE+MTYP)+.5D0*(BMOMT(KK)+STRS(
1      IND+IADD+KK))*DRN
    STRS(KI)=ROT(KK)
  END DO
END IF
C
C.... STORE LOCAL INTERNAL FORCES
C
IF (NOPT.EQ.1) THEN
  STRS(IND+2)=AFORCE(2)
  RETURN
ELSE IF (NOPT.EQ.2) THEN
  N=IND+2
ELSE IF (NOPT.EQ.3) THEN
  N=IND
END IF

STRS(N)=AFORCE(2)
STRS(N+1)=AFORCE(4)
STRS(N+2)=AFORCE(6)
DO I=2,6
  STRS(N+1+I)=BMOMT(I)
END DO

RETURN

100 FORMAT (2I10)

END

```

```
FUNCTION LOFIX (I)  
LOFIX=I  
RETURN  
END
```

```

SUBROUTINE PLOTTER ( NUMPNT, XARRAY, YARRAY, CLABEL, ULABEL, M, N )
C
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.....  A PLOTTING SUBROUTINE FOR THE LINE PRINTER]]
C.....  DUE TO THE COLUMN AND ROW WIDTHS, THIS SUBROUTINE MUS
C.....  DO QUITE A BIT OF TRUNCATING, HENCE THE ACCURACY OF THE P
C.....  IS RELATIVELY CRUDE.  HOWEVER, THE MAIN ADVANTAGE OF USIN
C.....  THIS SUBROUTINE IS THE SPEED OF THE LINE PRINTER OUTPUT.
C.....  ALSO NO SPECIAL HANDLING INSTRUCTIONS ARE REQUIRED.
C.....  .....
C.....  .
C.....  . NUMPNT  IS THE NUMBER DATA POINTS TO BE PLOTTED
C.....  .          (NUMPNT MUST NOT EXCEED 500, BUT IF IT
C.....  .          ONLY THE FIRST 500 POINTS ARE CONSIDER
C.....  .
C.....  . XARRAY  IS THE NAME OF THE ARRAY CONTAINING THE X-C
C.....  .
C.....  . YARRAY  IS THE NAME OF THE ARRAY CONTAINING THE Y-C
C.....  .
C.....  . CLABEL  IS THE GRAPH LABEL
C.....  .
C.....  . ULABEL  IS THE UNITS LABEL
C.....  .
C.....  . XMIN   IS THE MINIMUM X-VALUE FOR THE PLOTTING RAN
C.....  .
C.....  . XMAX   IS THE MAXIMUM X-VALUE FOR THE PLOTTING RAN
C.....  .
C.....  . YMIN   IS THE MINIMUM Y-VALUE FOR THE PLOTTING RAN
C.....  .
C.....  . YMAX   IS THE MAXIMUM Y-VALUE FOR THE PLOTTING RAN
C.....  .
C.....  .....
C.....  NOTE... A) THE SUBROUTINE PAGE EJECTS TO A NEW PAGE TO B
C.....  PLOTS, HOWEVER IT DOES NOT PAGE EJECT WHEN FI
C.....  AND TWO LINES REMAIN AT THE BOTTOM OF THE PAG
C.....  (THESE REMAINING LINES MAY BE USED TO IDENTIF
C.....  GRAPHS, IF WISHED, IN THE CALLING PROGRAM.)
C.....  B) THE PROGRAM AUTOMATICALLY PLOTS OVER A WHOLE
C.....  C) THE X AND Y ARRAYS REMAIN INTACT IN THE SUBRO
C.....  AND ARE RETURNED EXACTLY AS SENT.
C.....  D) IF MORE THAN ONE POINT EXISTS AT A LOCATION,
C.....  AN M-CHARACTER IS PLACED AT THAT POINT.
C.....  E) IF SOME POINTS ARE OUT OF THE RANGE OF THE GR
C.....  THEY ARE PLOTTED ON THE BORDERS. (HENCE THE
C.....  POINTS ARE INVALID.)
C.....  BEGINNING OF ACTUAL SUBROUTINE PLOTTER...
C
  INCLUDE 'ADJUST.COM'
  DIMENSION CHAR(101), XARRAY(NT), YARRAY(NA), IX(501), IY(501),
*    YVAL(3), CLABEL(10), ULABEL(10)
  DATA BLANK,STAR,YPNT,XPNT,DOT,PNTM/1H,,1H*,1HY,1HX,1H.,1HM/
C
C.....  ... IF NUMPNT>500, REDEFINE AS 500 ...
C
  IF (NUMPNT.GT.500) NUMPNT=500

  DO JDUM=1,501
    IX(JDUM)=1
    IY(JDUM)=1
  END DO
C
  XMIN=XARRAY(1)
  XMAX=XARRAY(1)
  YMIN=YARRAY(1)
  YMAX=YARRAY(1)

  DO IZ=1,NUMPNT
    IF (XARRAY(IZ).GT.XMAX) XMAX=XARRAY(IZ)
    IF (XARRAY(IZ).LT.XMIN) XMIN=XARRAY(IZ)
    IF (YARRAY(IZ).GT.YMAX) YMAX=YARRAY(IZ)
    IF (YARRAY(IZ).LT.YMIN) YMIN=YARRAY(IZ)
  END DO

  NUMBER=51
  FLNUMB=NUMBER

```

```

C
YVAL(1)=YMAX
YVAL(2)=(YMAX+YMIN)/2.D0
YVAL(3)=YMIN
C
C..... SETTING UP X AND Y INTEGER ARRAYS .....
C
DO JJ=1,NUMPNT
C
IF (YMAX-YMIN.EQ.0.) GO TO 7
IY(JJ)=(YARRAY(JJ)-YMIN)/(YMAX-YMIN)*(FLNUMB-1.D0)+.5D0
IF (IY(JJ) .GT. NUMBER) IY(JJ)=NUMBER
IF (IY(JJ)) 6,6,8
6 IY(JJ)=1
C
IF (XMAX-XMIN.GT.0.) GO TO 8
7 WRITE (6,901)
GO TO 50
8 IX(JJ)=(XARRAY(JJ)-XMIN)/(XMAX-XMIN)*(100.D0)+.5D0
IF (IX(JJ) .GT. 101) IX(JJ)=101
IF (IX(JJ) .LE. 0) IX(JJ)=1
C
END DO
C
C..... SORTING INTO DESCENDING IY-ARRAY ORDER .....
C
LIM=NUMPNT-1
13 INT=1
DO I=1,LIM
IF (IY(I+1) .LE. IY(I)) CYCLE
TEMP=IX(I+1)
IX(I+1)=IX(I)
IX(I)=TEMP
TEMP2=IY(I+1)
IY(I+1)=IY(I)
IY(I)=TEMP2
INT=I
END DO
IF (INT-1) 16,17,16
16 LIM=INT-1
GO TO 13
17 CONTINUE
C
C..... END OF SORTING LOOP
C..... *****
C..... FINDING X-AXIS & Y-AXIS POSITIONS, IF PRESENT]
C
IXKODE=0
IYKODE=0
C
C..... ...FIRST THE Y-AXIS...
C
IF (YMAX*YMIN) 18,18,21
18 IXAXIS=(YMAX/(YMAX-YMIN))*FLNUMB+.5D0
IF (IXAXIS .LE. 0) IXAXIS=1
IXKODE=1
C
C..... ...THEN THE X-AXIS...
C
21 IF (XMAX*XMIN) 22,22,25
22 IYAXIS=(-XMIN)/(XMAX-XMIN)*101.D0+.5D0
IF (IYAXIS .LE. 0) IYAXIS=1
IYKODE=1
25 CONTINUE
C
IF (M .EQ. -1) THEN
WRITE(6,908) (CLABEL(J), J=1,10)
ELSE IF (M .GT. N) THEN
WRITE(6,902) (CLABEL(I), I=1,10), (ULABEL(J), J=1,7)
ELSE
WRITE (6,907) (CLABEL(I), I=1,10), (ULABEL(J), J=1,10)
END IF
C
C..... *****
C..... BEGINNING OF ACTUAL PLOT LOOP
C

```

```

KNTR=1
KNTYAX=0
C
DO 49 LL=1,NUMBER
C
C....      SET UP PLOT LINE
C....      PLUG IN X-AXIS IF PRESENT (IXKODE = 1)
C
IF (IXKODE) 29,29,26
26 IF (LL-IXAXIS) 29,27,29
27 DO IJK=1,101
CHAR(IJK)=XPNT
END DO
GO TO 37
C
29 CHAR(1)=DOT
CHAR(101)=DOT
C
DO K=2,100
IF (LL-1) 30,33,30
30 IF (LL-NUMBER) 31,33,31
C
C....      BLANK OUT ROW
C
31 CHAR(K)=BLANK
END DO

GO TO 35
C
C....      IF FIRST OR LAST LINE, ADD BORDER DOTS
C
33 DO KK=2,100
CHAR(KK)=DOT
END DO
C
C....      IF IYKODE KNTR SET, ADD Y AXIS LABEL
C
35 IF (IYKODE .GT. 0) CHAR(IYAXIS)=YPNT
37 CONTINUE
C
IF (IY(KNTR)-((NUMBER+1)-LL)) 44,38,44
C
38 ICOL=IX(KNTR)
IF (CHAR(ICOL)-STAR) 39,40,39
39 IF (CHAR(ICOL)-PNTM) 41,40,41
40 CHAR(ICOL)=PNTM
GO TO 42
C
41 CHAR(ICOL)=STAR
C
42 IF (KNTR-NUMPNT) 43,44,43
43 KNTR=KNTR+1
IF (IY(KNTR-1)-IY(KNTR)) 44,38,44
C
C....      PRINTOUT COMPLETED ROW
C
44 IF (LL-1) 48,48,45
45 IF (LL-26) 46,48,46
46 IF (LL-NUMBER) 47,48,47
C
47 WRITE (6,903) CHAR
GO TO 49
C
48 KNTYAX=KNTYAX+1
WRITE (6,904) YVAL(KNTYAX),DOT,DOT,CHAR
C
49 CONTINUE
C
C....      END OF PLOTTING LOOP
C....      *****
C
XMID=(XMIN+XMAX)/2.D0
WRITE (6,905) XMIN,XMID,XMAX
C
IF (M.EQ. -1) GO TO 54
C

```

```

50 WRITE (6,906)
   RETURN
C
C
54 WRITE (6,909) (ULABEL(J), J=1,10)
   RETURN

901 FORMAT (1H1//51X,19HALL VALUES ARE ZERO//)
902 FORMAT ('I', 30X, 10A4, 10X, 8A4, //)
903 FORMAT (20X,101A1)
904 FORMAT(4X,D11.4,3X,2A1,101A1)
905 FORMAT(2(20X,1H.,49X,1H.,49X,1H.),15X,D11.4,39X,D11.4,40X,D11.4
   1./)
906 FORMAT (59X,14HTIME - SECONDS)
907 FORMAT ('I', 30X, 10A4, 10X, 10A4, //)
908 FORMAT ('I', 50X, 10A4, //)
909 FORMAT ( 50X, 10A4 )
   END

```



```

SUBROUTINE READIN (SMASS,E,XC,YC,ZC,NODE,XT,YT,ZT,MESHIN,INMESH,NO
IDDIS,ANGLE)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
INCLUDE 'ADJUST.COM'
INCLUDE 'SIZE.COM'
INTEGER PRITYP,SECTYP
DIMENSION XC(NXC), YC(NYC), ZC(NZC), E(12,NE), NODE(14,NIX),
1   NODDIS(NNODIS), XT(NXO), YT(NX1), ZT(NDICOS), JJ(7),
2   SMASS(NSMASS), ANGLE(9,ANGLE), NNREAD(8), MESHIN(
3   NMESHN), INMESH(NINMEH), TMASS(4)
INCLUDE 'DYNAM.COM'
INCLUDE 'OUTPA.COM'
INCLUDE 'CONTRL.COM'
CS
CS  COMMON STATEMENTS ADDED TO ACCOMODATE I/O DECLARATIONS
CS
COMMON /DEVS/LUTRM, LUIN, LUOUT, LUPLT
COMMON /LABELS/NAINP,NAOUT,NAPLT
DATA PRITYP,SECTYP/IHP,IHS/
WRITE (6,901) NNODE,NPRI,NAXOR,NELE,NUMMAT,NUMDIS,MXSTEP,NDGREE,DE
ILT,NODMAX
NODET=NNODE+NAXOR
READ (5,902) (KONTRL(I),I=1,16)
WRITE (6,903) KONTRL(1)
WRITE (6,904) KONTRL(2)
WRITE (6,905) KONTRL(3)
WRITE (6,906) KONTRL(4)
WRITE (6,907) KONTRL(5)
WRITE (6,908) KONTRL(6)
WRITE (6,913) KONTRL(7)
WRITE (6,915) KONTRL(8)
WRITE (6,916) KONTRL(9)
WRITE (6,917) KONTRL(10)
WRITE (6,918) KONTRL(11)
WRITE (6,9180) KONTRL(12)
WRITE (6,9181) KONTRL(13)
WRITE (6,9182) KONTRL(14)
WRITE (6,9183) KONTRL(15)
WRITE (6,9184) KONTRL(16)
DO I=1,NUMMAT
  READ (5,919) MTYPE
  READ (5,920) (E(J,MTYPE),J=1,12)
  WRITE (6,921) MTYPE,(E(J,MTYPE),J=1,12)
END DO
C
C....  READ AND PRINT NODAL POINT DATA
C
NEQ=NPRI*NDGREE
DO LEN=1,NEQ
  SMASS(LEN)=0.D0
END DO

IF (KONTRL(1).EQ. 1) THEN
  WRITE(6,923)
ELSE
  WRITE(6,922)
END IF

WRITE (6,924)
INTPRI=0
NCOUNT=0
INTSEC=NPRI
5 READ (5,926) MESH,NODTYP,XORD,YORD,ZORD,(TMASS(K),K=1,4)
NCOUNT=NCOUNT+1
IF (NNODE.EQ.NPRI) GO TO 8
IF (TMASS(1)) 6,7,8
6 WRITE (6,927) MESH,TMASS(1)
STOP
7 IF (NODTYP.EQ.PRITYP) GO TO 8
IF (NODTYP.EQ.SECTYP) GO TO 9
IF(KONTRL(7)) 9,9,8
8 INTPRI=INTPRI+1
INT=INTPRI
GO TO 10

```

```

9 INTSEC=INTSEC+1
  INT=INTSEC
10 WRITE (6,925) INT,MESH,NODTYP,XORD,YORD,ZORD,(TMASS(K),K=1,4)
  INMESH(MESH)=INT
  MESHIN(INT)=MESH
  XC(INT)=XORD
  YC(INT)=YORD
  ZC(INT)=ZORD
C
  IF (TMASS(1).NE.0.) THEN
    LEN=NDGREE*(INT-1)+1
    SMASS(LEN)=TMASS(1)
    SMASS(LEN+1)=TMASS(1)
    SMASS(LEN+2)=TMASS(1)
    SMASS(LEN+3)=TMASS(2)
    SMASS(LEN+4)=TMASS(3)
    SMASS(LEN+5)=TMASS(4)
  END IF

  IF (NNODE-NCOUNT) 12,13,5
12 WRITE (6,928) MESH
  STOP
13 CONTINUE
  IF (NAXOR.LE.0) GO TO 15
14 READ (5,926) MESH,NODTYP,XORD,YORD,ZORD
  NCOUNT=NCOUNT+1
  INTSEC=INTSEC+1
  WRITE (6,925) INTSEC,MESH,NODTYP,XORD,YORD,ZORD
  INMESH(MESH)=INTSEC
  MESHIN(INTSEC)=MESH
  XC(INTSEC)=XORD
  YC(INTSEC)=YORD
  ZC(INTSEC)=ZORD
  IF (NODET-NCOUNT) 12,15,14
C
15 CONTINUE
C
C.... READ AND PRINT ELEMENT NODES
C
  WRITE (6,929)
  N=0
16 N=N+1
  READ (5,930) M,(NNREAD(LEN),LEN=1,8),(NODE(K,N),K=9,14)
  DO LEN=1,8
    NLEN=NNREAD(LEN)
    IF (NLEN.EQ.0) THEN
      NODE(LEN,N)=0
    ELSE
      NODE(LEN,N)=INMESH(NLEN)
    END IF
  END DO

  WRITE (6,931) N,(NNREAD(I),I=1,8),(NODE(K,N),K=9,14),(NODE(J,N),J=
11,4),NODE(8,N)
  IF (N.LT.NELE) GO TO 16
C
C.... PRINT NODAL DATA IN COORDINATE SYSTEM NOT INPUT
C.... PASS ON TO REMAING CODE NODAL DATA IN GLOBAL COORDINATES
C.... REGARDLESS OF WHICH SYSTEM WAS CHOSEN FOR INPUT
C
  IF (KONTRL(1).EQ.1) GO TO 30
  WRITE (6,932)
  WRITE (6,924)
C
  SIGN=1.D0
C
19 DO 29 KK=1,NELE
C
C.... CHECK FOR PVP ELEMENTS (ETYPE=4)
C
  IF (NODE(10,KK).EQ.4) GO TO 22
C
  N1=NODE(1,KK)
  N2=NODE(2,KK)
  N3=NODE(3,KK)
  N4=NODE(4,KK)

```

```

C
C..... CHECK FOR PLATE ELEMENTS (ETYPE=5)
C
      IF (NODE(10,KK).EQ.5) GO TO 24
C
C..... CHECK FOR PRIMARY OR SECONDARY NODES
C
      IF (N3 .EQ. 0) THEN
        N3=N1
        NODE(3,KK)=N1
      END IF

      IF (N4 .EQ. 0) THEN
        N4=N2
        NODE(4,KK)=N2
      END IF

      XT(N1)=XC(N1)-XC(N3)*SIGN
      YT(N1)=YC(N1)-YC(N3)*SIGN
      ZT(N1)=ZC(N1)-ZC(N3)*SIGN
      XT(N2)=XC(N2)-XC(N4)*SIGN
      YT(N2)=YC(N2)-YC(N4)*SIGN
      ZT(N2)=ZC(N2)-ZC(N4)*SIGN
      XT(N3)=XC(N3)
      YT(N3)=YC(N3)
      ZT(N3)=ZC(N3)
      XT(N4)=XC(N4)
      YT(N4)=YC(N4)
      ZT(N4)=ZC(N4)
C
      GO TO 29
C
22  N7=NODE(7,KK)
     N8=NODE(8,KK)
     DO II=1,3
       NI=NODE(II,KK)
       NJ=NODE(II+3,KK)
       XT(NI)=XC(NI)-XC(N7)*SIGN
       YT(NI)=YC(NI)-YC(N7)*SIGN
       ZT(NI)=ZC(NI)-ZC(N7)*SIGN
       XT(NJ)=XC(NJ)-XC(N8)*SIGN
       YT(NJ)=YC(NJ)-YC(N8)*SIGN
       ZT(NJ)=ZC(NJ)-ZC(N8)*SIGN
     END DO
     XT(N7)=XC(N7)
     YT(N7)=YC(N7)
     ZT(N7)=ZC(N7)
     XT(N8)=XC(N8)
     YT(N8)=YC(N8)
     ZT(N8)=ZC(N8)
C
      GO TO 29
C
24  N5=NODE(5,KK)
     N6=NODE(6,KK)
     IF (N4.NE.0) GO TO 25
     N4=N1
     NODE(4,KK)=N1
25  IF (N5.NE.0) GO TO 26
     N5=N2
     NODE(5,KK)=N2
26  IF (N6.NE.0) GO TO 27
     N6=N3
     NODE(6,KK)=N6
27  DO II=1,3
     NI=NODE(II,KK)
     NJ=NODE(II+3,KK)
     XT(NI)=XC(NI)-XC(NJ)*SIGN
     YT(NI)=YC(NI)-YC(NJ)*SIGN
     ZT(NI)=ZC(NI)-ZC(NJ)*SIGN
     XT(NJ)=XC(NJ)
     YT(NJ)=YC(NJ)
     ZT(NJ)=ZC(NJ)
  END DO
C
29  CONTINUE

```

```

      WRITE (6,933) (KK,MESHIN(KK),XT(KK),YT(KK),ZT(KK),KK=1,NNODE)
C
      IF (SIGN) 31,33,33
C
30      WRITE (6,934)
      WRITE (6,924)
C
      SIGN=-1.0
C
      GO TO 19
31      DO KK=1,NNODE
          XC(KK)=XT(KK)
          YC(KK)=YT(KK)
          ZC(KK)=ZT(KK)
      END DO
33      CONTINUE
C
C....  READ DISPLACEMENT NODE CODES
C
      NPDIS=0
      IF (NUMDIS.EQ.0) GO TO 38
      WRITE (6,935)
      DO I=1,NUMDIS
          READ (5,936) (JJ(K),K=1,7),(ANGLE(LS,I),LS=1,6)
          JJ=JJ(1)
          JJ(1)=INMESH(JJ1)
          DO J=2,7
              IF (JJ(J).GT.1) NPDIS=NPDIS+1
          END DO
          CALL INCODE (NODDIS(I),JJ,3,7)
          CALL DECOD (NODDIS(I),JJ,3,7)
          IF (JJ(7).GT.NNODE) WRITE (6,947) JJ(7)
          JJ1=JJ(7)
          JJ(7)=MESHIN(JJ1)
          DO LS=1,3
              IF (ANGLE(LS,I).NE.0.) GO TO 36
          END DO
          ANGLE(1,I)=1.D0
          ANGLE(5,I)=1.D0
36      CALL CROSS (ANGLE(1,I),ANGLE(4,I),ANGLE(7,I),DUM,CMAG,0)
          WRITE (6,937) I,JJ(7),(JJ(J),J=1,6),(ANGLE(LS,I),LS=1,6)
      END DO
38      CONTINUE
C
      RETURN

901  FORMAT (1H0,4X,37HNUMBER OF PRIMARY AND SECONDARY NODES,110/5X,37H
1NUMBER OF PRIMARY NODES      ,110/5X,37HNUMBER OF AXIS ORI
2ENTATION NODES      ,110/5X,37HNUMBER OF ELEMENTS
3 ,110/5X,37HNUMBER OF MATERIALS      ,110/5X,37HNUMBER
4 OF FIXED NODES      ,110/5X,37HMAXIMUM NUMBER OF INCREM
5ENTS      ,110/5X,37HNO. OF DEGREES OF FREEDOM PER NODE ,110/
65X,37HTIME INCREMENT      ,1PD16.5/5X,37HMAXIMUM
7 NODE NUMBER      ,110,/)
902  FORMAT (16I5)
903  FORMAT (/1X,10HKONTRL(1)=,I5,10X,47H0 - SECONDARY NODES INPUT IN G
1LOBAL COORDINATES/,26X,46H1 - SECONDARY NODES INPUT IN LOCAL COOR
2DINATES)
904  FORMAT (/7X,4H(2)=,I5,10X,34H1 - OMIT PRINTING OF UOUT AND SOUT)
905  FORMAT (/7X,4H(3)=,I5,10X,44H0 - USE EIGEN TO FIND PRINCIPLE INER
1TIA AXES/,26X,54H1 - PRINCIPLE INERTIA AXES ARE THE SAME AS GLOBA
2L AXES/,26X,54H2 - INITIAL ORIENTATION DATA FOR RIGID BODIES IS I
3NPUT)
906  FORMAT (/7X,4H(4)=,I5,14X,34HNUMBER OF SLIDING INTERFACE PLANES)
907  FORMAT (/7X,4H(5)=,I5,10X,45HNEWMARK - BETA TEMPORAL INTEGRATION P
1ARAMETER/,26X,48H0 - EXPLICIT ( ONLY OPTION CURRENTLY AVAILABLE )
2)
908  FORMAT (/7X,4H(6)=,I5,10X,'NO. OF SECONDARY BODIES FOR CONTACT')
913  FORMAT (/7X,4H(7)=,I5,10X,'0 - DEFAULT NODE TYPE S SECONDARY NODE'
/,26X,'1 - DEFAULT NODE TYPE P PRIMARY NODE')
915  FORMAT (/7X,4H(8)=,I5,10X,37H0 - FRACTION CRITICAL ELEMENT DAMPING
1 /,26X,41H1 - STIFFNESS PROPORTIONAL GLOBAL DAMPING )
916  FORMAT (/7X,4H(9)=,I5,10X,'0-RECTANGULAR BEAM CROSS SECTION', 23
1H(ONE INTEGRATION POINT)/,26X,17H1-CROSS SECTIONAL,24HGEOMETRIES T
20 BE READ IN)
917  FORMAT (/6X,5H(10)=,I5,10X,'RESTART CONTROL')

```

```

918 FORMAT(1H0,5X,5H(11)=,I5,10X,32HQ ARRAY OUTPUT AT THIS TIME STEP)
9180 FORMAT(/,6X,5H(12)=,I5,10X,12H - NOT USED)
9181 FORMAT(/,6X,5H(13)=,I5,10X,13H0 - NO ACTION/,26X,52H1 - NUMERICAL
1 INTEGRATION DATA TO BE READ IN BY ICIF)
9182 FORMAT(/,6X,5H(14)=,I5,10X,'0 - ALL NODES ASSUMED INITIALLY IN CON
1TACT WITH SEATBACK',/26X,'1 - ARRAY OF INITIAL DISTANCES FROM SEA
2TBACK IS INPUT')
9183 FORMAT(/,6X,5H(15)=,I5,10X,13H0 - NO ACTION/,26X,24H1 - SPINIF DA
1TA REQUIRED)
9184 FORMAT(/,6X,5H(16)=,I5,10X,13H0 - NO ACTION/,26X,31H1 - RETRACTIO
1N / --- SIMULATION)
919 FORMAT (I5,D10.4)
920 FORMAT (6D10.4)
921 FORMAT (1H0,4X,3HMAT,I3,10X/3X,8HE-MATRIX,1P6D18.5/11X,1P6D18.5)
922 FORMAT (///,1X,41HNODAL DATA AS INPUT IN GLOBAL COORDINATES)
923 FORMAT (///,1X,40HNODAL DATA AS INPUT IN LOCAL COORDINATES)
924 FORMAT (30H0 INTERNAL NO. MESH NO.,39H X-ORDINATE Y-ORD
1INATE Z-ORDINATE)
925 FORMAT (12X,I4,8X,I4,4X,A1,3F13.3,1P4D15.5)
926 FORMAT (I5,4X,A1,7D10.4)
928 FORMAT (1H0,20HNODAL POINT ERROR N=,I5)
930 FORMAT (15I5)
931 FORMAT (1X,I8,5X,19I5)
932 FORMAT (///,1X,31HNODAL DATA IN LOCAL COORDINATES)
933 FORMAT (12X,I4,8X,I4,3F13.3)
934 FORMAT (///,1X,32HNODAL DATA IN GLOBAL COORDINATES)
935 FORMAT (1H0,10X,18HDISPLACEMENT NODES/13X,8HMESH NO.,5X,9HCONDITIO
1N)
936 FORMAT (I4,6I1,6D10.4)
937 FORMAT (5X,I5,5X,I4,5X,6I2,3X,6F10.5)
947 FORMAT(1H0,5X,25HWARNING DISP. NODE NUMBER ,I5)
927 FORMAT(///,1H ,24HNODAL MASS ERROR, NODE= ,I5,10X,5HMASS=,D20.4)
929 FORMAT(1H0/,2X,
16SHELEMENT NO. N1 N2 N3 N4 N5 N6 N7 COOR MTYP ET
2YP ,5X,26H N1 N2 N3 N4 COOR )

```

END

```

SUBROUTINE READOU(UOUT,SOUT,NPOUT,GLABEL,INMESH,NODE,DP,NSEC,IMPB)

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

INCLUDE 'ADJUST.COM'
INCLUDE 'OUTPA.COM'
INCLUDE 'SIZE.COM'
C$
C$ COMMON STATEMENTS ADDED TO ACCOMODATE I/O DECLARATIONS
C$
COMMON /DEVS/LUTRM, LUIN, LUOUTP, LUPLT
COMMON /LABELS/NAINP,NAOUT,NAPLT
INTEGER UOUT,SOUT,LUTRM, LUIN, LUOUTP, LUPLT

DIMENSION DP(NDEF)
DIMENSION INMESH(NINMEH), JJ(7), SOUT(NSOUT), NPOUT(2,NNPOUT),
1      NODE(14,NIX), UOUT(NUOUT)
DIMENSION IMPB(NISEC)
DIMENSION GLABEL( 20,NGLABE )

WRITE (6,939)
WRITE (6,940) NPFREQ, NPRU, NPRS, NPIC, NANG, NPSEC
IF (NPRU.EQ.0) GO TO 43
READ (5,941) (UOUT(IQ), (GLABEL(IQ,JQ), IQ=1,17), JQ=1,NPRU)
WRITE(6,942) (UOUT(JQ), (GLABEL(IQ,JQ), IQ=1,17), JQ=1,NPRU)
IFLAG=0

DO I=1,NPRU
  CALL DECOD (UOUT(I),JJ,10,4)
  IF(JJ(3).EQ.3) IFLAG=1
  JJ(4)=INMESH(JJ(4))
  IF ( JJ(4) .GT. NNODE ) GO TO 55
  DO II=1,4
    JJ(6-II)=JJ(5-II)
  END DO
  JJ(1)=JJ(5)
  CALL INCODE (UOUT(I),JJ,10,4)
END DO

43 IF (NPRS.EQ.0) GO TO 45
READ (5,941) (SOUT(JQ), (GLABEL(IQ,NPRU+JQ), IQ=1,17), JQ=1,NPRS)
WRITE(6,943) (SOUT(JQ), (GLABEL(IQ,NPRU+JQ), IQ=1,17), JQ=1,NPRS)
DO I=1,NPRS
  CALL DECOD (SOUT(I),JJ,10,4)
  IF ( JJ(4) .GT. NELE ) GO TO 56
END DO
45 CONTINUE
IF (NPIC.LE.0) GO TO 47
WRITE (6,944)
DO K=1,NPIC
  READ (5,945) (NPOUT(I,K),I=1,2)
  IF(NPOUT(2,K).EQ.5) IFLAG=1
  WRITE (6,946) (NPOUT(I,K),I=1,2)
END DO
47 IF (NPFREQ.LE.0) NPFREQ=1

NUMHAL=NNODE/2
K=0
DO JE=1,NELE
  KJE=NODE(3,JE)-NODE(4,JE)
  IF (NODE(10,JE).EQ.4) KJE=NODE(7,JE)-NODE(8,JE)
  IF (KJE.LT.0) KJE=-KJE
  IF (KJE.GT.K) K=KJE
END DO
IF(NANG.EQ.0) GO TO 52
READ(5,960) AM,(DP(I),DP(I+NANG),DP(I+2*NANG),DP(I+3*NANG),I=1,
1      NANG)
WRITE(6,961) AM,(DP(I),DP(I+NANG),DP(I+2*NANG),DP(I+3*NANG),I=1,
1      NANG)
NANG=AM*NANG
52 CONTINUE
MUD=(K+1)*NDGREE-1
C
IF(NSEC.LE.0) RETURN
DO 50 I=1,NSEC
  NA=6*(I-1)

```

```

50 READ(5,950) II,(IMPB(NA+K),K=1,6)
   RETURN
55 WRITE (6,948) JJ(4)
   GO TO 60
56 WRITE (6,949) JJ(4)
60 CONTINUE

   STOP
939 FORMAT (1H0//5X,11HOUTPUT CODE)
940 FORMAT (1H0,9X,7HNPFREQ=,I10/10X,7HNPRU =,I10/10X,7HNPRS =,
1    I10/10X,7HNPIC =,I10/,10X,7HNANG =,I10/,10X,
2    7HNPSEC =,I10)
941 FORMAT (I10, 17A4)
942 FORMAT ('0',5X,'UOUT - SELECTIVE NODAL DISPLACEMENT, VELOCITY, ',
* 'AND ACCELERATION OUTPUT',/(6X,I10,10X,17A4))
943 FORMAT ('0',5X,'SOUT - SELECTIVE ELEMENT STRESS ARRAY OUTPUT'
* (6X,I10,10X,17A4))
944 FORMAT ('0',5X,'NPOUT PICTURE OUTPUT'/10X,'AT STEP',10X,'CODE')
945 FORMAT (2I10)
946 FORMAT (//10X,I7,10X,I4)
948 FORMAT ('0',5X,'WARNING UOUT NODE NUMBER =',I5)
949 FORMAT ('0',5X,'WARNING SOUT ELEMENT NUMBER =',I5)
950 FORMAT(7I5)
960 FORMAT(8F10.0)
961 FORMAT(1H0,5X,'DEFORMATION AMPLIFICATION FACTOR =',F10.2,/,5X,
1 'WIDTH',5X,'X-ROTATION',5X,'Y-ROTATION',5X,'Z-ROTATION',
2 /,(4X,D12.5,3(3X,D12.5)))

END

```

```

      SUBROUTINE ROTATE (R,V,KODE)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.... THIS ROUTINE APPLIES THE ROTATION MATRIX R TO THE VECTOR
C.... V AS INDICATED BY THE PARAMETER KODE
C.... KODE = 1 V = R * V
C.... KODE = 2 V = R(TRAN) * V
C.... IF KODE IS NEGATIVE V IS A SIX ELEMENT VECTOR
C
      COMMON /MATRIX/ NAM,NB,NR
      DIMENSION R(6),V(6),T(6)
C
      IKODE=IABS(KODE)
      NAM = 3*3
      NB = 3*1
      NR = 3*1
      GO TO (1,2), IKODE
C
1 CALL GMPRD (R,V,T,3,3,1)
  IF (KODE.GT.0) GO TO 3
  CALL GMPRD (R,V(4),T(4),3,3,1)
  GO TO 3
C
2 CALL GTPRD (R,V,T,3,3,1)
  IF (KODE.GT.0) GO TO 3
  CALL GTPRD (R,V(4),T(4),3,3,1)
C
3 DO 4 I=1,3
  V(I)=T(I)
4 CONTINUE
  IF (KODE.GT.0) GO TO 6
  DO 5 I=4,6
  V(I)=T(I)
5 CONTINUE
C
6 RETURN
  END

```



```

SUBROUTINE ROTE (THETA,PHI,PSI,XO,YO,ZO,XP,YP,CMAX,NNODE)

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.... THETA= ROTATION ABOUT THE Y AXIS
C.... PHI= ROTATION ABOUT THE X AXIS
C.... PSI=ROTATION ABOUT Z AXIS
Cbrc VARIABLE NAMES CHANGED BY DJP
C
INCLUDE 'ADJUST.COM'
DIMENSION XO(NNODE), YO(NNODE), ZO(NNODE), CA(2), CB(2), CG(2),
1      XP(NNODE), YP(NNODE), CMAX(6)
THR=THETA*DATAN(1.D0)/45.D0
PHIR=PHI*DATAN(1.D0)/45.D0
PSIR=PSI*DATAN(1.D0)/45.D0
CA(1)=DCOS(THR)*DCOS(PSIR)-DSIN(THR)*DSIN(PHIR)*DSIN(PSIR)
CA(2)=-DSIN(THR)*DSIN(PHIR)*DCOS(PSIR)-DCOS(THR)*DSIN(PSIR)
CB(1)=DCOS(PHIR)*DSIN(PSIR)
CB(2)=DCOS(PHIR)*DCOS(PSIR)
CG(1)=-DSIN(THR)*DCOS(PSIR)-DCOS(THR)*DSIN(PHIR)*DSIN(PSIR)
CG(2)=-DCOS(THR)*DSIN(PHIR)*DCOS(PSIR)+DSIN(THR)*DSIN(PSIR)
CMAX(1)=XO(1)
CMAX(2)=XO(1)
CMAX(3)=YO(1)
CMAX(4)=YO(1)
CMAX(5)=ZO(1)
CMAX(6)=ZO(1)

DO I=1,NNODE
  IF (XO(I).GE.CMAX(1)) CMAX(1)=XO(I)
  IF (XO(I).LE.CMAX(2)) CMAX(2)=XO(I)
  IF (YO(I).GE.CMAX(3)) CMAX(3)=YO(I)
  IF (YO(I).LE.CMAX(4)) CMAX(4)=YO(I)
  IF (ZO(I).GE.CMAX(5)) CMAX(5)=ZO(I)
  IF (ZO(I).LE.CMAX(6)) CMAX(6)=ZO(I)
  ZO(I)=-ZO(I)
  XP(I)=XO(I)*CA(1)+YO(I)*CB(1)+ZO(I)*CG(1)
  YP(I)=XO(I)*CA(2)+YO(I)*CB(2)+ZO(I)*CG(2)
  ZO(I)=-ZO(I)
END DO

RETURN
END

```

```

      SUBROUTINE SFRCIN (L,ND,XC,YC,ZC,IX,E,UD,FINT,STRS,STRAIN,STRESS,N
      IUMEL,INDEX,SMASS,EUI,EUJ,TRAI,TRAJ,EH,AL,IEIGEN,INMESH)
Cbrcc
Cbrcc IT APPEARS THIS SUBROUTINE NEVER GETS CALLED!
Cbrcc
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.... THIS ROUTINE CALCULATES THE INTERNAL FORCE IN AN AXIAL 3-D SPRING
C.... AND TRANSFERS THE FORCE AND MOMENT DUE TO THE SPRING FORCE TO
C.... THE ASSOCIATED RIGIB BODY
C.... TRAI AND TRAJ CONTAIN R-ZERO AND R-ZERO-BAR FOR NODES I & J
C
      COMMON/ALINE/NS,LEL(40),UCP(120),OMEGA(360),EUL(360)
      INCLUDE 'ADJUST.COM'
      INCLUDE 'MATRIX.COM'
      INCLUDE 'NUMINT.COM'
      INCLUDE 'CONTRL.COM'

      DOUBLE PRECISION FINT(NFINT)

      DIMENSION XC(NXC), YC(NYC), ZC(NZC), IX(14,NIX), E(12,NE), UD(NX1)
      1  ,STRAIN(NSTRAI), STRESS(NSTRES), STRS(NSTRS), INDEX(
      2  NINDEX), EUI(9), EUJ(9), TRAI(6), TRAJ(6), EH(3), OMEGI(
      3  3,3), OMEGJ(3,3), TEMP1(3), TEMP2(3), TEMP3(3), TEMP4(3)
      4  ,AL(NAL), DISP(2), SMASS(NSMASS), INMESH(NINMEH)
C
      IF(KONTRL(16),EQ.0) GO TO 30
C
C CHECK IF ELEMENT L IS AN ELEMENT ASSOCIATED WITH THE RIGHT OR LEFT
C SHOULDER RIGID BODIES OR RIBS.
C
      ISET=0
      NAM = 3*3
      NB = 3*1
      NR = 3*1
Cbrcc
Cbrcc NOTE: WHERE IS NS SET? SINCE THIS SUBROUTINE ISN'T CALLED, WE'RE
Cbrcc NOT SURE. BROWSER SHOWS THAT THE COMMON BLOCK ISN'T USED ANYWHERE
Cbrcc ELSE.
Cbrcc
      DO 31 I=1,NS
      IF (L.NE.LEL(I)) GO TO 31
      ISET=1
      IND=I
      31 CONTINUE
C
      30 N1=IX(1,L)
      N2=IX(2,L)
      N3=IX(3,L)
      N4=IX(4,L)
C
      ISKIP=0
      JSKIP=0
C
C.... CHECK FOR PRIMARY OR SECONDARY NODES
C
      IF (N3.EQ.N1) ISKIP=1
      IF (N4.EQ.N2) JSKIP=1
C
C.... SET UP OMEGA TRANSFORMATION MATRICES
C
      IF (ISKIP.EQ.1) GO TO 1
      OMEGI(1,1)=0.
      OMEGI(1,2)=TRAI(6)
      OMEGI(1,3)=-TRAI(5)
      OMEGI(2,1)=-TRAI(6)
      OMEGI(2,2)=0.
      OMEGI(2,3)=TRAI(4)
      OMEGI(3,1)=TRAI(5)
      OMEGI(3,2)=-TRAI(4)
      OMEGI(3,3)=0.
      1 IF (JSKIP.EQ.1) GO TO 2
      OMEGJ(1,1)=0.
      OMEGJ(1,2)=TRAJ(6)
      OMEGJ(1,3)=-TRAJ(5)
      OMEGJ(2,1)=-TRAJ(6)

```

```

OMEGJ(2,2)=0.D0
OMEGJ(2,3)=TRAJ(4)
OMEGJ(3,1)=TRAJ(5)
OMEGJ(3,2)=-TRAJ(4)
OMEGJ(3,3)=0.D0
C
IF(KONTRL(16).EQ.0) GO TO 32
IF(ISET.EQ.0) GO TO 32
N=9*(IND-1)+1
K=1
DO 33 J=1,3
DO 33 I=1,3
OMEGA(N)=OMEGJ(I,J)
EUL(N)=EUI(K)
N=N+1
33 K=K+1
32 CONTINUE
C.... TRANSFORM R-ZERO-BAR TO GLOBAL SYSTEM
C
2 IF (ISKIP.EQ.1) GO TO 3
TEMP1(1)=TRAI(4)
TEMP1(2)=TRAI(5)
TEMP1(3)=TRAI(6)
3 IF (JSKIP.EQ.1) GO TO 4
TEMP2(1)=TRAJ(4)
TEMP2(2)=TRAJ(5)
TEMP2(3)=TRAJ(6)
4 IF (ISKIP.EQ.1) GO TO 5
CALL GMPRD (EUI,TEMP1,TEMP3,3,3,1)
5 IF (JSKIP.EQ.1) GO TO 6
CALL GMPRD (EUI,TEMP2,TEMP4,3,3,1)
C
C.... DETERMINE SECONDARY NODE GLOBAL DISPLACEMENTS
C
6 XX=XC(N2)-XC(N1)
YY=YC(N2)-YC(N1)
ZZ=ZC(N2)-ZC(N1)
N1N=(N1-1)*ND
N2N=(N2-1)*ND
N3N=(N3-1)*ND
N4N=(N4-1)*ND
IF (ISKIP.EQ.1) GO TO 7
DIX=UD(N3N+1)+TEMP3(1)-TRAI(1)
DIY=UD(N3N+2)+TEMP3(2)-TRAI(2)
DIZ=UD(N3N+3)+TEMP3(3)-TRAI(3)
UD(N1N+1)=DIX
UD(N1N+2)=DIY
UD(N1N+3)=DIZ
GO TO 8
7 DIX=UD(N3N+1)
DIY=UD(N3N+2)
DIZ=UD(N3N+3)
8 IF (JSKIP.EQ.1) GO TO 9
DIX=UD(N4N+1)+TEMP4(1)-TRAJ(1)
DJY=UD(N4N+2)+TEMP4(2)-TRAJ(2)
DIJZ=UD(N4N+3)+TEMP4(3)-TRAJ(3)
C
IF(KONTRL(16).EQ.0) GO TO 34
IF(ISET.EQ.0) GO TO 34
M=3*(IND-1)+1
UCP(M)=DJX
UCP(M+1)=DJY
UCP(M+2)=DIJZ
34 CONTINUE
C
UD(N2N+1)=DJX
UD(N2N+2)=DJY
UD(N2N+3)=DIJZ
GO TO 10
9 DJX=UD(N4N+1)
DJY=UD(N4N+2)
DIJZ=UD(N4N+3)
10 IF (IEIGEN.EQ.1) GO TO 19
C
C.... DETERMINE ELEMENT STRAIN
C

```

```

DIJX=DJX-DIX
DIJY=DJY-DIY
DIJZ=DJZ-DIZ
DX=XX+DIJX
DY=YY+DIJY
DZ=ZZ+DIJZ
AL2=AL(L)*AL(L)
FAC=2.D0*(XX*DIJX+YY*DIJY+ZZ*DIJZ)+DIJX*DIJX+DIJY*DIJY+DIJZ*DIJZ
ALN2=AL2+FAC
ALN=DSQRT(ALN2)
STREH=FAC/(AL(L)+ALN)/AL(L)
C
C.... DETERMINE NEW DIRECTION COSINES OF ELEMENT MU=EH
C
EH(1)=DX/ALN
EH(2)=DY/ALN
EH(3)=DZ/ALN
C
C.... DETERMINE LOCAL INTERNAL FORCES
C
DISP(1)=ALN
DISP(2)=STREH
MTYP=IX(9,L)
I1=1
NOPT=IX(10,L)
CALL LOCFRC (L,DISP,TEMP1,TEMP2,TEMP3,E,INDEX,I1,STRAIN,STRESS,STR
IS,SMAS,IX,AL(L),MTYP,NOPT,NUMEL,ZC,INMESH)
FIXH=TEMP2(1)
FJXH=TEMP2(2)
C
C.... IF(FJXH.NE.0.) WRITE(6,201) L,FJXH
C.... IF(FJXH.NE.0.) WRITE(6,202) L,FJXH
C
IF (FJXH.EQ.0.) GO TO 19
C
C.... ACCUMULATE INTERNAL ELEMENT FORCE IN FINT IN GLOBAL SYSTEM
C
DO 11 M=1,3
TEMP1(M)=EH(M)*FIXH
TEMP2(M)=EH(M)*FJXH
FINT(N3N+M)=FINT(N3N+M)+TEMP1(M)
11 FINT(N4N+M)=FINT(N4N+M)+TEMP2(M)
C
C.... ACCUMULATE MOMENT DUE TO INTERNAL FORCE IN FINT IN BODY SYSTEM
C
IF (ISKIP.EQ.1) GO TO 12
CALL GTPRD (EUI,TEMP1,TEMP3,3,3,1)
CALL GTPRD (OMEGI,TEMP3,TEMP1,3,3,1)
GO TO 14
12 DO 13 M=1,3
13 TEMP1(M)=0.D0
14 IF (JSKIP.EQ.1) GO TO 15
CALL GTPRD (EUJ,TEMP2,TEMP4,3,3,1)
CALL GTPRD (OMEGJ,TEMP4,TEMP2,3,3,1)
GO TO 17
15 DO 16 M=1,3
16 TEMP2(M)=0.D0
17 I=0
DO 18 M=4,6
I=I+1
FINT(N3N+M)=FINT(N3N+M)+TEMP1(I)
18 FINT(N4N+M)=FINT(N4N+M)+TEMP2(I)
19 RETURN
C
END

```

```

SUBROUTINE SLIDER (NOP,FINT,SMAS,UD2,UD,TIME,NCP,NASN,INMESH,
2DICOS,ALPHA,UP,UP2,NPNO,SEATK,UPOLD,SEATWK,SEATEX,UD1,UP1,VDAMP)
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C INCLUDE 'ADJUST.COM'
C INCLUDE 'MATRIX.COM'
C INCLUDE 'CONTRL.COM'
COMMON /SEATM/ USEAT,VSEAT,ASEAT
DIMENSION NCP(10,NNCP), X1(3), X2(3), X3(3), V1(3), V2(3), UP2(
1 NUP2), UD2(NAO), ALPHA(3,3,NALPHA), UD(NX1),
2 UP(NUP), NASN(NNASN), INMESH(NINMEH), DICOS(3,NDICOP),
3 NPNO(NNPNO), SEATK(2,NSEATK), UPOLD(NUPOLD), UD1(NVO),
4 UP1(NUP1), VDAMP(NVDAMP)
DIMENSION SMAS(NSMAS)
DOUBLE PRECISION FINT(NFINT)
C
C DIMENSION DELTY0(50)
C
C DELTY0 IS THE ARRAY OF INITIAL DISTANCES FROM THE SEATBACK
C
C IF (TIME.GT.0.) GO TO 9
C
C DO 7 I=1,NOP
C READ (5,901) NPNO(I),NASN(I),(DICOS(J,I),J=1,3),(SEATK(J,I),J=1,
1 2),VDAMP(I)
C IF (NASN(I).GT.0) GO TO 1
C READ (5,902) (NCP(J,I),J=1,2)
C GO TO 2
1 I STOP=NASN(I)
C READ (5,902) (NCP(J,I),J=1,ISTOP)
2 READ (5,903) (X1(J),J=1,3),(X2(J),J=1,3),(X3(J),J=1,3)
C WRITE (6,904) NPNO(I),NASN(I)
C WRITE (6,905) (DICOS(J,I),J=1,3)
C WRITE (6,906) (SEATK(J,I),J=1,2),VDAMP(I)
C WRITE (6,907)
C IF (NASN(I).GT.0) GO TO 3
C WRITE (6,902) (NCP(J,I),J=1,2)
C GO TO 4
3 I STOP=NASN(I)
C WRITE (6,902) (NCP(J,I),J=1,ISTOP)
4 WRITE (6,908) (X1(J),J=1,3),(X2(J),J=1,3),(X3(J),J=1,3)
C
C
C IF REQUIRED, READ IN THE INITIAL DISTANCES FROM THE SEATBACK OF
C NODES WHICH MAY CONTACT SEATBACK. IT IS ASSUMED THAT THE PLANE I = 1
C IS THE SEATBACK.
C
C IF(I.GT.1) GO TO 1007
C IF(KONTRL(14).EQ.1) GO TO 1002
C DO 1001 KK=1,50
1001 DELTY0(KK)=0.
C GO TO 1007
1002 IF(NASN(I).GT.0) GO TO 1003
C NCONT=NCP(2,I)-NCP(1,I)+1
C GO TO 1004
1003 NCONT=NASN(I)
1004 READ(5,1005) (DELTY0(KK),KK=1,NCONT)
1005 FORMAT(8F10.3)
C WRITE(6,1006) (DELTY0(KK),KK=1,NCONT)
1006 FORMAT('0 ARRAY OF INITIAL DISTANCES FROM SEATBACK'/(10F10.3))
C
1007 TIMAG=0.
C DO 5 J=1,3
C V1(J)=X2(J)-X1(J)
C TIMAG=TIMAG+V1(J)*V1(J)
5 V2(J)=X3(J)-X1(J)
C UN1=V1(2)*V2(3)-V2(2)*V1(3)
C UN2=V1(3)*V2(1)-V2(3)*V1(1)
C UN3=V1(1)*V2(2)-V2(1)*V1(2)
C UNMAG=SQRT(UN1*UN1+UN2*UN2+UN3*UN3)
C TIMAG=SQRT(TIMAG)
C ALPHA(1,1,I)=UN1/UNMAG
C ALPHA(1,2,I)=UN2/UNMAG
C ALPHA(1,3,I)=UN3/UNMAG
C

```

```

C.... REPLACE MESH NODE NO. WITH INTERNAL NODE NO.
C
  ISTOP=NASN(I)
  NPNO(I)=INMESH(NPNO(I))
  IF (ISTOP.EQ.0) ISTOP=2
  DO 6 LEN=1,ISTOP
    NMESH=NCP(LEN,I)
  6   NCP(LEN,I)=INMESH(NMESH)
      WRITE (6,909)
      WRITE (6,902) (NCP(LEN,I),LEN=1,ISTOP)
C
  7 CONTINUE
C
  K=3*NOP
  DO 8 J=1,K
    UP(J)=0.
    UP1(J)=0.
  8   UP2(J)=0.
      RETURN
C
  9 SEATWK=0.D0
    SEATEX=0.D0
C
C TEMP MOD 7-6-84 - NO CALLS TO ICIF FROM SLIDER - SEAT MOTION, GIVEN
C BY USEAT,VSEAT AND ASEAT, OBTAINED FROM FREEFD THROUGH
C COMMON/ SEATM /
C
C IF(KONTRL(13).EQ.0) GO TO 10
C CALL ICIF (TIME,ASEAT,0)
C CALL ICIF (TIME,VSEAT,1)
C CALL ICIF (TIME,USEAT,2)
C 10 DO 13 I=1,NOP
C
  10 DO 13 I=1,NOP
    KN=6*(NPNO(I)-1)
    DO 12 J=1,3
      K=3*(I-1)+J
      UPOLD(K)=UP(K)
      UP2(K)=ASEAT*DICOS(J,I)
      UP1(K)=VSEAT*DICOS(J,I)
      UP(K)=USEAT*DICOS(J,I)
    12 UD(KN+J)=UP(K)
    13 CONTINUE
C
  DO 22 I=1,NOP
C
  C
    ISKIP=0
    IF (NASN(I).EQ.0) GO TO 14
    ISTART=1
    IEND=NASN(I)
    ISKIP=1
    GO TO 15
  14 ISTART=NCP(1,I)
    IEND=NCP(2,I)
  15 K=3*(I-1)
    IABE=0
    DO 22 J=ISTART,IEND
      IABE=IABE+1
      NN=J
      IF (ISKIP.EQ.1) NN=NCP(J,I)
      KN=6*(NN-1)
      DELTND=0.
      DELTND=DELTND+DELTND*(IABE)
      ABE=DELTND
      VELNOR=0.
      ACCNOR=0.
      DO 16 L=1,3
        ACCNOR=ACCNOR+(UD2(KN+L)-UP2(K+L))*ALPHA(1,L,I)
        VELNOR=VELNOR+(UD1(KN+L)-UP1(K+L))*ALPHA(1,L,I)
  16 DELTND=DELTND+(UD(KN+L)-UP(K+L))*ALPHA(1,L,I)
      IF (DELTND.GT.0.) GO TO 22
      IF (VELNOR.GT.0.) GO TO 22
      KP=6*(NPNO(I)-1)
      SEATF=(SEATK(1,I)+SEATK(2,I)*DELTND*DELTND)*DELTND
      IF (KONTRL(8).EQ.1) GO TO 17
      FDAMP=2.0D0*VDAMP(I)*DSQRT(SEATK(1,I)*SMASST(KN+1))*VELNOR

```

```

GO TO 18
17 FDAMP=VDAMP(I)*SEATK(1,I)*VELNOR
18 SEATF=SEATF+FDAMP
DO 20 L=1,3
  SEATEX=SEATEX-(SEATF+SMASS(KN+L)*ACCNOR)*(UP(K+L)-UPOLD(K+L))*
1  ALPHA(1,L,I)
  FINT(KP+L)=FINT(KP+L)-SEATF*ALPHA(1,L,I)
20 FINT(KN+L)=FINT(KN+L)+SEATF*ALPHA(1,L,I)
C.... 19 FORCD(KN+L)=FORCD(KN+L)+V2(L)
C
  SEATWK=SEATWK+SEATF*DELTND
22 CONTINUE
  RETURN
901 FORMAT (2I5,6E10.0)
902 FORMAT (10I5)
903 FORMAT (3E10.0)
904 FORMAT (/5X,5HPLANE,I5,5X,28HNO OF ASSOCIATED SPACE NODES,I5)
905 FORMAT (10X,17HDIRECTION COSINES,10X,8HX - AXIS,12X,8HY - AXIS,12X
1,8HZ - AXIS,/,28X,1P3E20.4)
906 FORMAT (5X,36HELASTIC SEATBACK STIFFNESS LINEAR=,1PE15.4,10X,6HC
1UBIC=,1PE15.4,10X,14HDAMPING COEF =,1PE12.4)
907 FORMAT (5X,22HASSOCIATED SPACE NODES)
908 FORMAT (5X,25HPLANE NODES X1,X2, AND X3,12X,11HX-COMPONENT,9X,11HY
1-COMPONENT,9X,11HZ-COMPONENT,/,30X,2HX1,1P3E20.4,/,30X,2HX2,1P3E20
2.4,/,30X,2HX3,1P3E20.4,/)
909 FORMAT (5X,43HINTERNAL NODE NO. OF ASSOCIATED SPACE NODES)
910 FORMAT (3(5X,1P3E20.4,/,)3X,1P6E20.4)
911 FORMAT (10X,1P6E20.4)
END

```

```

SUBROUTINE SOLVE (NODDIS,SMASS,XC,YC,ZC,IX,E,XO,X1,V,AO,A1,FORCD,F
1INT,EULCO,DICOS,INMESH,AL,INDEX,STRS,STRAIN,STRESS,IPT,FEXOLD,
2NCP,NASN,DICOSP,ALPHA,UP,UP2,NPNO,SEATK,UPOLD,VDAMP,UP1,VOLD)

```

```

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

```

```

INCLUDE 'ADJUST.COM'
INCLUDE 'SIZE.COM'
INCLUDE 'CONTRL.COM'
INCLUDE 'DYNAM.COM'
INCLUDE 'OUTPA.COM'
INCLUDE 'ARRAY.COM'
INCLUDE 'INDEX.COM'

```

```

DOUBLE PRECISION FINT(NFINT)

```

```

DIMENSION XO(NXO), X1(NX1), V(NVO), AO(NAO), A1(NA1), FORCD(
1  NFORCD), NODDIS(NNODIS), SMASS(NSMASS), XC(NXC), YC(
2  NYC), IX(14,NIX), E(12,NE), JJ(7), AT(3), DICOS(9,
3  NDICOS), INMESH(NINMEH), ZC(NZC), EULCO(3,3,NBLAMB)
DIMENSION INDEX(NINDEX), STRS(NSTRS), STRAIN(NSTRAI), STRESS(
1  NSTRES), AL(NAL), FEXOLD(NFEXOD), NCP(10,NNCP), NASN(
2  NNASN), DICOSP(3,3,NDICOP), ALPHA(3,3,NALPHA), UP(NUP),
3  UP2(NUP2), NPNO(NNPNO), SEATK(2,NSEATK), UPOLD(NUPOLD),
4  UP1(NUP1), VDAMP(NVDAMP), IPT(NIPT)
DIMENSION VOLD(NVOLD)
DIMENSION ABIRD(600),VBIRD(600)

```

```

C
C.... DIMENSION STATEMENT NEEDED FOR EXTERNAL ENERGY CALCULATION
C.... WHEN PRESCRIBED DEGREES OF FREEDOM ARE USED (20 P.D.O.F.)

```

```

C
C      DIMENSION XPRED(20), ITPD(20), FROLD(20)

```

```

C
C.... CONTROL VARIABLES FOR INTEGRATION

```

```

C
ICON=0
NSLIDP=KONTRL(4)
BETA=KONTRL(5)
IB=KONTRL(6)*6
MEQ=NUMNP*NDGREE
NEQ=NPRI*NDGREE
IEIGEN=0
SEATWK=0.D0
SEATEX=0.D0
IF (BETA.NE.0.) GO TO 54

```

```

C
C.... ZERO DEPENDENT VARIABLES

```

```

C
DO KK=1,20
  XPRED(KK)=0.D0
  FROLD(KK)=0.D0
END DO
DO I=1,MEQ
  XO(I)=0.D0
  X1(I)=0.D0
END DO
IND=INDEX(NELE+1)-1
IEND=NUMMAT+2
LLSTRS=IND+IEND
WRITE (6,901) LLSTRS
DO M=1,IEND
  N=IND+M
  STRS(N)=0.D0
END DO

```

```

C
C.... COMPUTE INTEGRATION PARAMETERS

```

```

C
TIME=0.D0
NTSTEP=0
C1=(.5D0-BETA)*DELT*DELT
C2=BETA*DELT*DELT
C3=DELT/2.D0

```

```

C
C.... INITIAL CONDITIONS

```

```

C
CALL FREEFD (2,NUMDIS,NODDIS,NUMNP,NDGREE,XC,YC,ZC,X1,V,AO,FORCD,I
INMESH,BETA)

```

```

C

```



```

C.... CALCULATE INITIAL KINETIC ENERGY
C
  UKE=0.D0
  DO LES=1,NEQ
    UKE=UKE+SMASS(LES)*V(LES)*V(LES)
  END DO
  IEE=INDEX(NELE+1)+NUMMAT+1
  STRS(IEE)=.5D0*UKE
C
  IF (NSLIDP.NE.0)
  1 CALL SLIDER (NSLIDP,FINT,SMASS,A1,X1,TIME,NCP,NASN,
  2   INMESH,DICOSP,ALPHA,UP,UP2,NPNO,SEATK,UPOLD,
  3   SEATWK,SEATEX,V,UP1,VDAMP)
  CALL UPDATE ( NPRI,NDGREE,DELT,X1,XO,V,A1,EULCO(1,1,1),BETA )
  CALL FRCIN (NELE,NDGREE,XC,YC,ZC,IX,E,X1,FINT,STRS,
  1   STRAIN,STRESS,IPT,INDEX,EULCO,SMASS,DICOS,NPRI,
  2   AL,IEIGEN,INMESH)
  WRITE (6,902)
  WRITE (6,903) (LEN,AL(LEN),LEN=1,NELE)
  GO TO 7
  ENTRY RESOLV
  BETA=KONTRL(5)
  MEQ=NUMNP*NDGREE
  NEQ=NPRI*NDGREE
  IEIGEN=0
  NSLIDP=KONTRL(4)
  SEATWK=0.D0
  SEATEX=0.D0
  C1=(.5D0-BETA)*DELT**2
  C2=BETA*DELT*DELT
  C3=DELT*.5D0
C
C.... INTEGRATION LOOP
C
  7 ITER=0
  NTSTEP=NTSTEP+1
  TIME=TIME+DELT
  DO I=1,NEQ
    FORCD(I)=0.D0
    A1(I)=AO(I)
  END DO
  CALL FREEFD (1,NUMDIS,NODDIS,NUMNP,NDGREE,XC,YC,ZC,X1,V,AO,
  1   FORCD,INMESH,BETA)
  ITER=ITER+1
C
C.... ALTER PRESCRIBED DISPLACEMENTS
C.... ZERO ALL ACC. AND VEL. CORRESPONDING TO PRESCRIBED DISPLACEMENT
C
  NPDOF=0
  IF (NUMDIS.LE.0) GO TO 18
  DO I=1,NUMDIS
    CALL DECOD (NODDIS(I),JJ,3,7)
    J7=JJ(7)
    ML=NDGREE*(J7-1)
    DO 14 J=1,NDGREE
      ML=ML+1
      IF (JJ(J)-1) 14,12,11
    11 NPDOF=NPDOF+1
      XPRED(NPDOF)=XO(ML)
      XO(ML)=X1(ML)
      ITPD(NPDOF)=ML
      GO TO 13
    12 X1(ML)=0.D0
      XO(ML)=0.D0
    13 V(ML)=0.D0
      AO(ML)=0.D0
      A1(ML)=0.D0
      ABIRD(ML)=0.D0
      VBIRD(ML)=0.D0
      FORCD(ML)=0.D0
    14 CONTINUE
  END DO
C
C.... COMPUTE NEW DISPLACEMENTS
C
  18 I=0

```

```

19 I=I+1
   IF (I.LE.NEQ) THEN
     IF (SMASS(I).NE.0.) THEN
       X1(I)=X0(I)+V(I)*DEL T+C1*AO(I)
       GO TO 19
     END IF
     I=I+5
     GO TO 19
   END IF
C
  CALL UPDATE (NPRI,NDGREE,DEL T,X1,X0,V,A1,EULCO,BETA)
  CALL FRCIN (NELE,NDGREE,XC,YC,ZC,IX,E,X1,FINT,STRS,
1    STRAIN,STRESS,IPT,INDEX,EULCO,SMASS,DICOS,NPRI,AL,
2    IEIGEN,INMESH)
  IF (NSLIDP.NE.0)
1  CALL SLIDER (NSLIDP,FINT,SMASS,A1,X1,TIME,NCP,NASN,
2    INMESH,DICOSP,ALPHA,UP,UP2,NPNO,SEATK,UPOLD,
3    SEATWK,SEATEX,V,UP1,VDAMP)

  MJ=0
23 DO I=1,3
    IF (SMASS(I+MJ).EQ.0.) GO TO 26
    A1I=(FORCD(I+MJ)-FINT(I+MJ))/SMASS(I+MJ)
    A1(I+MJ)=A1I
  END DO
  M1=MJ+4
  M2=MJ+5
  M3=MJ+6
  M4=MJ+3
  AT(1)=(FORCD(M1)-FINT(M1)+(SMASS(M2)-SMASS(M3))*V(M2)*V(M3))
1    /SMASS(M1)
  AT(2)=(FORCD(M2)-FINT(M2)+(SMASS(M3)-SMASS(M1))*V(M3)*V(M1))
1    /SMASS(M2)
  AT(3)=(FORCD(M3)-FINT(M3)+(SMASS(M1)-SMASS(M2))*V(M1)*V(M2))
1    /SMASS(M3)
  DO I=1,3
    A1(M4+I)=AT(I)
  END DO
26 MJ=MJ+NDGREE
  IF (MJ.LT.NEQ) GO TO 23

  DO I=1,NEQ
    VOLD(I)=V(I)
    V(I)=V(I)+C3*(A1(I)+AO(I))
    AO(I)=A1(I)
  END DO
C
C.... COMPUTE KINETIC ENERGY AND EXTERNAL ENERGY
C
  IKE=INDEX(NELE+1)
  IEE=IKE+NUMMAT+1
  STRS(IKE)=0.D0
  I=0
36 I=I+1
  IF (I.GT.NEQ) GO TO 38
  IF (SMASS(I).EQ.0.) GO TO 37
  STRS(IKE)=STRS(IKE)+SMASS(I)*V(I)*V(I)
  GO TO 36
37 I=I+5
  GO TO 36
C
C.... CORRECT KINETIC ENERGY AND ADD EXTERNAL ENERGY FOR
C.... PRESCRIBED DISPLACEMENT DEGREES OF FREEDOM
C
38 IF (NPDOF.EQ.0) GO TO 41
  DO KK=1,NPDOF
    ML=ITPD(KK)
    STRS(IKE)=STRS(IKE)-SMASS(ML)*V(ML)*V(ML)
  END DO
  CALL FREEFD (3,NUMDIS,NODDIS,NUMNP,NDGREE,XC,YC,ZC,X1,V,A1,
1    FORCD,INMESH,BETA)
  DO KK=1,NPDOF
    ML=ITPD(KK)
    STRS(IKE)=STRS(IKE)+SMASS(ML)*V(ML)*V(ML)
    FRACT=FINT(ML)+SMASS(ML)*A1(ML)
    STRS(IEE)=STRS(IEE)+.5D0*(FRACT+FROLD(KK))*(X1(ML)-XPRED(KK))
  END DO

```

```

      FROLD(KK)=FRACT
    END DO
C
C.... CALCULATION OF EXTERNAL ENERGY WHEN SLIDING INTERFACES ARE USED
C
41 IF (NSLIDP.EQ.0) GO TO 51
   DO 50 LS=1,NSLIDP
     ISKIP=0
     IF (NASN(LS).EQ.0) GO TO 42
     ISTART=1
     IEND=NASN(LS)
     ISKIP=1
     GO TO 43
42  ISTART=NCP(1,LS)
     IEND=NCP(2,LS)
43  DO 49 J=ISTART,IEND
     NN=J
     IF (ISKIP.EQ.1) NN=NCP(J,LS)
C
C.... CHECK FOR NODES WHOSE MOTION IS PRESCRIBED BY MORE THAN
C.... ONE PLANE
C
   IF (LS.EQ.1) GO TO 47
   NCHECK=LS-1
   DO 46 LEN=1,NCHECK
     IF (NASN(LEN).EQ.0) GO TO 45
     JSTART=1
     JEND=NASN(LEN)
     DO LL=JSTART,JEND
       NKK=NCP(LL,LEN)
       IF (NN.EQ.NKK) GO TO 49
     END DO
     GO TO 46
45  NSTART=NCP(1,LEN)
     NEND=NCP(2,LEN)
     IF (NN.GE.NSTART.AND.NN.LE.NEND) GO TO 49
46  CONTINUE
C
47  KDOF=6*(NN-1)
     DO LES=1,3
       ML=KDOF+LES
       FORCD(ML)=FINT(ML)+SMASS(ML)*AO(ML)
     END DO
49  CONTINUE
50  CONTINUE
C
51 STRS(IKE)=.5D0*STRS(IKE)+.5D0*SEATWK
C
   WRKEXT=SEATEX
   DO KK=1,NEQ
     WRKEXT=WRKEXT+(FORCD(KK)+FEXOLD(KK))*(X1(KK)-XO(KK))
     FEXOLD(KK)=FORCD(KK)
     XO(KK)=X1(KK)
   END DO
C
   STRS(IEE)=STRS(IEE)+.5D0*WRKEXT
   NWRKP=MOD(NTSTEP,NPFREQ)
   IF (NWRKP.NE.0) GO TO 53
   IF (NSLIDP.EQ.0) GO TO 53
   WRITE (6,904) SEATWK,SEATEX
53 CONTINUE
C
   IF(NTSTEP.GT.1) GO TO 58
   NANG=(LUOUT-LDEF)/4
   IF (NANG.EQ.0) THEN
     AM = 0
   ELSE
     AM=DFLOAT(NAN/NANG)
   END IF
   NAN=NANG
58  CONTINUE
   IF(ICON.EQ.1) GO TO 70
   DO KK=1,NEQ
     VBIRD(KK)=V(KK)
     ABIRD(KK)=A1(KK)
   END DO

```

```

70  CONTINUE
    CALL OUTPUT(Q(LXC),Q(LYC),Q(LZC),Q(LIX),Q(LXI),
1      VBIRD,ABIRD,NUMNP,NPRI,NELE,NDGREE,NUMMAT,
2      Q(LINDEX),STRS,Q(LBLAMB),Q(LMESHN),Q(LINMEH),
3      Q(LUOUT),Q(LSOUT),Q(LNPOUT),Q(LGLABE),Q(LUU),Q(LSS),
4      Q(LT),Q(LA),Q(LPSU),Q(LNTYPE),Q(LAUX),Q(LIPT),AM)
    IF (NTSTEP.LT.MXSTEP) GO TO 7
C
C....  END OF LOOP
C
    RETURN
54 WRITE (6,905) BETA

    STOP
C
901 FORMAT (/5X,23HLENGTH OF STRS ARRAY=,I5)
902 FORMAT (/10X,23HORGINAL ELEMENT LENGTHS,/)
903 FORMAT(5(5X,I5,5X,D10.3))
904 FORMAT (5X,10HSEAT INWK=,1PD12.4,5X,10HSEAT EXWK=,1PD12.4)
905 FORMAT (1H0, 5X, 'BETA = ', D15.6)
911 FORMAT ( 16I5 )
    END

```

```

SUBROUTINE SPINIF(JE,JSTART,JEND,NB,FORCE,BMOMT,ZC,NZC,INMESH,
1      NINMEH)
C
C PERKIN - ELMER VERSION
C
C REPLACES PREVIOUS HSM SPINAL INJURY PREDICTION POSTPROCESSOR,
C INJCRI
C
C SPINIF DETERMINES THE SPINAL INJURY FUNCTION,
C SIF = FUNCTION( COMPUTED AXIAL COMPRESSION AND BENDING MOMENTS
C      PLUS YIELD CRITERIA BASED, IN PART, ON EXPERIMEN-
C      TALLY MEASURED VERTEBRAL LOAD-DEFORMATION DATA )
C
C JE = ELEMENT NUMBER
C JSTART = INFERIOR DISC ELEMENT FOR BOTTOM MOST VERTEBRAL LEVEL
C      CONSIDERED
C JEND = SUPERIOR ELEMENT FOR UPPERMOST VERTEBRAL LEVEL CONSIDERED
C NB = NO. VERTEBRAL LEVELS CONSIDERED
C FORCE = ELEMENT JE LOCAL NODAL FORCE ARRAY
C BMOMT = ELEMENT JE LOCAL NODAL MOMENT ARRAY
C
C PY,BMY,BMZ = AXIAL COMPRESSION, LATERAL BENDING MOMENT AND
C      A-P BENDING MOMENT, RESPECTIVELY, CORRESPONDING
C      TO YIELDING OF THE CORTICAL SHELL
C ISYM = SYMMETRY OPTION
C      = 0, SIMULATION IS SYMMETRIC ABOUT GLOBAL YZ PLANE
C
C ALL LOADS ARE SAMPLED EVERY NPFREQ STEPS, WITH A SAMPLED VALUE
C COMPUTED AS THE MEAN OF ALL VALUES DURING A SAMPLING INTERVAL,
C AND TAKEN TO OCCUR AT THE MIDPOINT OF THE INTERVAL.
C
C AUGUST - 1983
C
C IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
C DIMENSION FORCE(6),BMOMT(6),PY(20),BMY(20),BMZY(20),Z(20),
1      ZC(NZC),INMESH(NINMEH),PE(20),VYE(20),VZE(20),TE(20),
2      BMYE(20),BMZE(20),PSUM(20),VYSUM(20),VZSUM(20),TSUM(20),BMYSUM
3      (20),BMZSUM(20),P(20),VY(20),VZ(20),T(20),BMY(20),BMZ
4      (20),TP(20),TVY(20),TVZ(20),TT(20),TMY(20),TMZ(20),F1
5      (20),PF1(20),BMF1(20),TF1(20),F2(20),PF2(20),BMF2(20),
6      TF2(20),F(20),PF(20),BMF(20),TF(20),TEMP(20),ZLABEL
7      (10),PLOT1(10),PLOT2(10),PLOT3(10),PLOT4(10),PLOT5(10)
8      ,PLOT6(10),PLOT7(10),PLOT8(10),PLOT9(10)
C DIMENSION TEA(20),BMYEA(20),BMZEA(20),VZEA(20),VYEA(20),FA(20) RVCC
C
C COMMON/DYNAM/DEL T,TIMEND,MXSTEP,NTSTEP,TIME
C COMMON/OUTPA/NPRU,NPRS,NPFREQ
C COMMON/CONTRL/KONTRL(16) RVCC
C
C DATA N1/1/
C DATA ZLABEL/4HVERT, 4HEBRA, 4HLE, 4HVEL, 6*4H /
C DATA PLOT1/4H P /, 4HPY, 8*4H /
C DATA PLOT2/4HBM, 4HBM, 4HYY, 7*4H /
C DATA PLOT3/4HBMZ, 4HBM, 4HZY, 7*4H /
C DATA PLOT4/4HVV, 9*4H /
C DATA PLOT5/4HVZ, 9*4H /
C DATA PLOT6/4HT, 9*4H /
C DATA PLOT7/4H" SA, 4HGITT, 4HAL ", 4HPLA, 4HNE I, 4HNJUR,
1      4HYFU, 4HNCTI, 4HON, 4H /
C DATA PLOT8/4H" FR, 4HONTA, 4HL ", 4HPLAN, 4HEIN, 4HJURY,
1      4HFUN, 4HCTIO, 4HN, 4H /
C DATA PLOT9/4H* *, 4H* SP, 4HINAL, 4HINJ, 4HURY, 4HFUNC,
1      4HTION, 4H* *, 4H* *, 4H /
C
C
C
C INITIALIZATION
C
C N2=N1*NPFREQ
C IF(TIME.GE.DEL T) GO TO 30
C IF(JE.GT.JSTART) GO TO 40
C WRITE(6,2000)
2000 FORMAT(1H1,51H --- SPINIF ( SPINAL INJURY FUNCTION ) DATA ---)
C READ(5,1000) (PY(I),I=1,NB)
1000 FORMAT(16F5.2)
C READ(5,1000) (BMY(I),I=1,NB)

```

```

      READ(5,1000) (BMZY(I),I=1,NB)
      READ(5,1001) FACT,NSTART,ISYM
1001 FORMAT(D10.5,2I5)
      WRITE(6,2001) FACT,NSTART,ISYM
2001 FORMAT(1H0,19H FACT,NSTART,ISYM =,D10.5,2I5)
      WRITE(6,2002)
2002 FORMAT(1H0,4X,5HLEVEL,8X,6HZ0(CM),7X,10HYIELD LOAD,4X,
113HLAT YIELD MOM,2X,14HA-P YIELD MOM ./)
      DO 10 I=1,NB
        PY(I)=PY(I)*FACT
        BMYI(I)=BMYI(I)*FACT*2.0
        BMZY(I)=BMZY(I)*FACT*2.0
        Z(I)=ZC(INMESH(NSTART+I-1))
10      WRITE(6,2003) I,Z(I),PY(I),BMYI(I),BMZY(I)
2003 FORMAT(1H ,5X,I2,3X,4D15.4)
C
      DO 20 I=1,NB
        PSUM(I)=0.D0
        VYSUM(I)=0.D0
        VZSUM(I)=0.D0
        TSUM(I)=0.D0
        BMYSUM(I)=0.D0
        BMZSUM(I)=0.D0
        P(I)=0.D0
        VY(I)=0.D0
        VZ(I)=0.D0
        T(I)=0.D0
        BMY(I)=0.D0
        BMZ(I)=0.D0
        F1(I)=0.D0
        F2(I)=0.D0
20      F(I)=0.D0
        FREQ=FLOAT(NPFREQ)
        PER=FREQ*DELT
30      CONTINUE
        IF(JE.GT.JSTART) GO TO 40
        JV=0
C
C   LOADS ON INFERIOR ENDPLATE OF FIRST VERTEBRA CONSIDERED -
C   TYPICALLY L5 ( HUMAN ), L6 OR L7 ( BABOON )
C
C   SIGN CONVENTIONS
C
C   PI = AXIAL FORCE - LT/GT 0 - C/T
C   VYI = LOCAL Y SHEAR - LT/GT 0 - FLEX/EXT
C   VZI = LOCAL Z SHEAR - LT/GT 0 - RIGHT/LEFT LAT BENDING
C   TI = TORSION - LT/GT 0 - NEG/POS ROT ABOUT ANATOMICAL Z
C   BMYI = LOCAL Y MOMENT - LT/GT 0 - RIGHT/LEFT LAT BENDING
C   BMZI = LOCAL Z MOMENT - LT/GT 0 - FLEX/EXT
C
      PI=FORCE(2)
      VYI=FORCE(4)
      IF(ISYM.EQ.0) GO TO 35
      VZI=-FORCE(6)
      TI=BMOMT(2)
      BMYI=BMOMT(4)
35      BMZI=BMOMT(6)
      GO TO 999
40      JV=JV+1
      IF(JE.EQ.JEND) GO TO 56
C
C   LOADS ON SUPERIOR ENDPLATES
C
C   SIGN CONVENTIONS FOR PS,VYS,VZS,TS,BMYS AND BMZS SIMILAR TO
C   THOSE FOR INFERIOR ENDPLATE LOADS
C
      PS=-FORCE(1)
      VYS=FORCE(3)
      IF(ISYM.EQ.0) GO TO 45
      VZS=-FORCE(5)
      TS=BMOMT(1)
      BMYS=-BMOMT(3)
45      BMZS=-BMOMT(5)
C
C   DETERMINE EQUILIBRIUM VALUES OF INTERNAL LOADS
C   EG. P ( EQUILIBRIUM ) = 0.5 * ( PI + PS )

```

```

C
PE(JV)=0.5D0*(PI+PS)
C WRITE(6,4105)
IF(PE(JV).GT.0.D0) PE(JV)=0.D0
C WRITE(6,4105)
VYE(JV)=0.5D0*(VYI+VYS)
C WRITE(6,4105)
IF(ISYM.EQ.0) GO TO 50
VZE(JV)=0.5D0*(VZI+VZS)
TE(JV)=0.5D0*(TI+TS)
BMYE(JV)=0.5D0*(BMYI+BMYS)
50 BMZE(JV)=0.5D0*(BMZI+BMZS)
C
C LOADS ON REMAINING INFERIOR ENDPLATES
C
PI=FORCE(2)
VYI=FORCE(4)
IF(ISYM.EQ.0) GO TO 55
VZI=-FORCE(6)
TI=BMOMT(2)
BMYI=BMOMT(4)
55 BMZI=BMOMT(6)
GO TO 999
56 PE(JV)=PI
IF(PE(JV).GT.0.D0) PE(JV)=0.D0
VYE(JV)=VYI
IF(ISYM.EQ.0) GO TO 57
VZE(JV)=VZI
TE(JV)=TI
BMYE(JV)=BMYI
57 BMZE(JV)=BMZI
C
C SUM LOADS
C
60 DO 70 I=1,NB
PSUM(I)=PSUM(I)+PE(I)
VYSUM(I)=VYSUM(I)+VYE(I)
IF(ISYM.EQ.0) GO TO 70
VZSUM(I)=VZSUM(I)+VZE(I)
TSUM(I)=TSUM(I)+TE(I)
BMYSUM(I)=BMYSUM(I)+BMYE(I)
70 BMZSUM(I)=BMZSUM(I)+BMZE(I)
C
C IF CURRENT TIME STEP IS NOT AN ENDPOINT OF A SAMPLING INTERVAL,
C RETURN
C
IF(NTSTEP.NE.N2) GO TO 999
TYME=0.5D0*(2.D0*DFLOAT(N1)-1.D0)*PER
C
DO 80 I=1,NB
PE(I)=PSUM(I)/FREQ
VYE(I)=VYSUM(I)/FREQ
VYEA(I)=DABS(VYE(I))
IF(ISYM.EQ.0) GO TO 71
VZE(I)=VZSUM(I)/FREQ
VZEA(I)=DABS(VZE(I))
TE(I)=TSUM(I)/FREQ
TEA(I)=DABS(TE(I))
BMYE(I)=BMYSUM(I)/FREQ
BMYEA(I)=DABS(BMYE(I))
71 BMZE(I)=BMZSUM(I)/FREQ
BMZEA(I)=DABS(BMZE(I))
IF(PE(I).GT.P(I)) GO TO 72
P(I)=PE(I)
TP(I)=TYME
72 IF(DABS(VYE(I)).LT.DABS(VY(I))) GO TO 73
VY(I)=VYE(I)
TVY(I)=TYME
73 IF(ISYM.EQ.0) GO TO 77
IF(DABS(VZE(I)).LT.DABS(VZ(I))) GO TO 74
VZ(I)=VZE(I)
TVZ(I)=TYME
74 IF(DABS(TE(I)).LT.DABS(T(I))) GO TO 75
T(I)=TE(I)
TT(I)=TYME
75 IF(DABS(BMYE(I)).LT.DABS(BMY(I))) GO TO 76

```

```

      BMY(I)=BMYE(I)
      TMY(I)=TYME
76  VZSUM(I)=0.D0
      TSUM(I)=0.D0
      BMYSUM(I)=0.D0
77  IF(DABS(BMZE(I)).LT.DABS(BMZ(I))) GO TO 78
      BMZ(I)=BMZE(I)
      TMZ(I)=TYME
78  PSUM(I)=0.D0
      VYSUM(I)=0.D0
      BMZSUM(I)=0.D0
80  CONTINUE
      N1=N1+1
C
C  UPDATE SPINAL INJURY FUNCTION
C
      DO 90 I=1,NB
      FP=DABS(PE(I))/PY(I)
      FMZ=DABS(BMZE(I))/BMZY(I)
      FTEMP=FP+FMZ
      IF(FTEMP.LT.F1(I)) GO TO 81
      F1(I)=FTEMP
      FA(I)=F1(I)
      PF1(I)=PE(I)
      BMF1(I)=BMZE(I)
      TF1(I)=TYME
81  FMY=0.D0
      IF(ISYM.EQ.0) GO TO 90
      FMY=DABS(BMYE(I))/BMYI(I)
      FTEMP=FP+FMY
      IF(FTEMP.LT.F2(I)) GO TO 82
      F2(I)=FTEMP
      PF2(I)=PE(I)
      BMF2(I)=BMYE(I)
      TF2(I)=TYME
82  FTEMP=FP+DMAX1(FMY,FMZ)
      IF(FTEMP.LT.F(I)) GO TO 90
      F(I)=FTEMP
      FA(I)=F(I)
      PF(I)=PE(I)
      BMF(I)=BMYE(I)
      IF(FMZ.GE.FMY) BMF(I)=BMZE(I)
      TF(I)=TYME
90  CONTINUE
C
C          RVCC
C IF COLOR CODING IS DESIRED, WRITE RV DATA TO LU 11.          RVCC
      IF(KONTRL(12).EQ.1) THEN          RVCC
        WRITE(11,5000) (PE(I),I=1,9)          RVCC
        WRITE(11,5001) (PE(I),I=10,17)          RVCC
        WRITE(11,5000) (TEA(I),I=1,9)          RVCC
        WRITE(11,5001) (TEA(I),I=10,17)          RVCC
        WRITE(11,5000) (BMYEA(I),I=1,9)          RVCC
        WRITE(11,5001) (BMYEA(I),I=10,17)          RVCC
        WRITE(11,5000) (BMZEA(I),I=1,9)          RVCC
        WRITE(11,5001) (BMZEA(I),I=10,17)          RVCC
        WRITE(11,5000) (VZEA(I),I=1,9)          RVCC
        WRITE(11,5000) (VZEA(I),I=10,17)          RVCC
        WRITE(11,5000) (VYEA(I),I=1,9)          RVCC
        WRITE(11,5001) (VYEA(I),I=10,17)          RVCC
        WRITE(11,5000) (FA(I),I=1,9)          RVCC
        WRITE(11,5001) (FA(I),I=10,17)          RVCC
5000  FORMAT(9D14.7)          RVCC
5001  FORMAT(8D14.7)          RVCC
      END IF          RVCC
C          RVCC
C  IF(NTSTEP.LT.MXSTEP) GO TO 999
C
C  OUTPUT
C
      WRITE(6,3000)
3000  FORMAT(1H1,75H ***** SPINAL INJURY LIKELIHOOD POSTPROCESSI
      ING OUTPUT ***** )
      WRITE(6,3001)
3001  FORMAT(1H-,71H ----- MAX COMPRESSIVE FORCES AND BENDING MO
      IMENTS ----- //
      228H P = MAX COMPRESSIVE FORCE//

```



```

356H BMZ = MAX LAT MOMENT - LT/GT 0 - RIGHT/LEFT LAT BENDING//
442H BMZ = MAX A-P MOMENT - LT/GT 0 - FLEX/EXT//
548H TP,TMY,TMZ = TIMES AT WHICH P,BMY AND BMZ OCCUR//
628H PY,BMY,BMZ = YIELD VALUES)
WRITE(6,3002)
3002 FORMAT(1H-,3X,5HLEVEL,6X,2HTP,10X,1HP,12X,2HPY,9X,3HTMY,
18X,3HBMZ,10X,4HBMYY,8X,3HTMZ,8X,3HBMZ,10X,4HBMZY/)
DO 100 I=1,NB
100 WRITE(6,3003) I,TP(I),P(I),PY(I),TMY(I),BMY(I),BMY(I),
1TMZ(I),BMZ(I),BMZY(I)
3003 FORMAT(1H,1X,15,4X,3(F10.6,1PD13.4,1PD13.4))
DO 110 I=1,NB
110 TEMP(I)=P(I)/PY(I)
CALL PLOTTER(NB,Z,TEMP,PLOT1,ZLABEL,-1,0)
DO 120 I=1,NB
120 TEMP(I)=BMZ(I)/BMZY(I)
CALL PLOTTER(NB,Z,TEMP,PLOT3,ZLABEL,-1,0)
IF(ISYM.EQ.0) GO TO 150
DO 130 I=1,NB
130 TEMP(I)=BMY(I)/BMY(I)
CALL PLOTTER(NB,Z,TEMP,PLOT2,ZLABEL,-1,0)
C
WRITE(6,3004)
3004 FORMAT(1H1,51H----- MAX SHEARS AND TORSION -----,
1//
239H VY = MAX Y SHEAR - LT/GT 0 - A/P SHEAR//
348H VZ = MAX LAT SHEAR - LT/GT 0 - RIGHT/LEFT SHEAR//
443H T = MAX TORSION - LT/GT 0 - +/- Z BAR ROT//
546H TVY,TVZ,TT = TIMES AT WHICH VY,VZ AND T OCCUR)
WRITE(6,3005)
3005 FORMAT(1H-,3X,5HLEVEL,6X,3HTVY,9X,2HVY,9X,3HTVZ,9X,2HVZ,9X,
12HTT,10X,1HT/)
DO 140 I=1,NB
140 WRITE(6,3006) I,TVY(I),VY(I),TVZ(I),VZ(I),TT(I),T(I)
3006 FORMAT(1H,1X,15,4X,3(F10.6,1PD13.4))
CALL PLOTTER(NB,Z,VY,PLOT4,ZLABEL,-1,0)
CALL PLOTTER(NB,Z,VZ,PLOT5,ZLABEL,-1,0)
CALL PLOTTER(NB,Z,T,PLOT6,ZLABEL,-1,0)
C
150 WRITE(6,3007)
3007 FORMAT(1H1,63H***** " SAGITTAL PLANE " INJURY FUNCTION *
1*****//
233H F1 = ( ABS(P/PY) + ABS(MZ/MZY) )//
344H TF1 = TIMES AT WHICH F1, PF1 AND BMF1 OCCUR)
WRITE(6,3008)
3008 FORMAT(1H-,3X,5HLEVEL,6X,3HTF1,9X,2HF1,10X,3HPF1,11X,2HPY,
110X,4HBMF1,9X,4HBMZY/)
DO 160 I=1,NB
160 WRITE(6,3009) I,TF1(I),F1(I),PF1(I),PY(I),BMF1(I),BMZY(I)
3009 FORMAT(1H,1X,15,4X,F10.6,5(1PD13.4))
CALL PLOTTER(NB,Z,F1,PLOT7,ZLABEL,-1,0)
IF(ISYM.EQ.0) GO TO 999
C
WRITE(6,3010)
3010 FORMAT(1H1,62H----- " FRONTAL PLANE " INJURY FUNCTION --
1-----//
233H F2 = ( ABS(P/PY) + ABS(MY/MYY) )//
344H TF2 = TIMES AT WHICH F2, PF2 AND BMF2 OCCUR)
WRITE(6,3011)
3011 FORMAT(1H-,3X,5HLEVEL,6X,3HTF2,9X,2HF2,10X,3HPF2,11X,2HPY,10X,
14HBMF2,9X,4HBMYY/)
DO 170 I=1,NB
170 WRITE(6,3012) I,TF2(I),F2(I),PF2(I),PY(I),BMF2(I),BMY(I)
3012 FORMAT(1H,1X,15,4X,F10.6,5(1PD13.4))
CALL PLOTTER(NB,Z,F2,PLOT8,ZLABEL,-1,0)
C
WRITE(6,3013)
3013 FORMAT(1H1,88H***** COMBINED " SAGITTAL " AND " FRONTAL "
1 PLANE INJURY FUNCTION *****//
253H F = ( ABS(P/PY) + MAX( ABS(MY/MYY) , ABS(MZ/MZY) ) )//
340H TF = TIMES AT WHICH F, PF AND BMF OCCUR)
WRITE(6,3014)
3014 FORMAT(1H-,3X,5HLEVEL,6X,2HTF,10X,1HF,12X,2HPF,11X,2HPY,10X,3HBMF,
110X,4HBMYY,9X,4HBMZY)
DO 180 I=1,NB
180 WRITE(6,3015) I,TF(I),F(I),PF(I),PY(I),BMF(I),BMY(I),BMZY(I)

```

```
3015 FORMAT(1H,1X,I5,4X,F10.6,6(1PD13.4))  
      CALL PLOT9(NB,Z,F,PLOT9,ZLABEL,-1.0)  
C  
999 RETURN  
      END
```

```

SUBROUTINE SUBTIM(HL,HS,ML,MS,SL,SS,CL,CS,DH,DM,DS,DC)
C
C SUBROUTINE TO SUBTRACT A SMALLER TIME FROM A LARGER TIME
C
  INTEGER*2 HL,HS,SHL,SHS,ML,MS,SML,SMS,SL,SS,SSL,SSS,
&    CL,CS,SCL,SCS,DH,DM,DS,DC
C
C PRESERVE THE CALLING VARIABLES
C
  SHL = HL
  SHS = HS
  SML = ML
  SMS = MS
  SSL = SL
  SSS = SS
  SCL = CL
  SCS = CS
C
C DO ALL ARITHMETIC WITH SAVED VARIABLES
C
  IF(SCL .LT. SCS) THEN
    DC = SCL + 100 - SCS
    SSL = SSL - 1
  ELSE
    DC = SCL - SCS
  END IF
  IF(SSL .LT. SSS) THEN
    DS = SSL + 60 - SSS
    SML = SML - 1
  ELSE
    DS = SSL - SSS
  END IF
  IF(SML .LT. SMS) THEN
    DM = SML + 60 - SMS
    SHL = SHL - 1
  ELSE
    DM = SML - SMS
  END IF
  DH = SHL - SHS
  RETURN
END

```

```

SUBROUTINE UPDATE (NPRI,NDGREE,DELT,X1,XO,V,A,BLAMB,BETA)
C
Cbrc
Cbrc CLEANED UP SOME CODE - DJP
Cbrc
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
INCLUDE 'ADJUST.COM'
COMMON /DYNAM/ NDUM(3), NTSTEP, TIME
DOUBLE PRECISION E1,E3,E31S,E32S
C
DIMENSION V(NVO), A(NA1), BLAMB(NNBLAM), E1(3), E3(3), TEMP1(3),
1 TEMP2(3), X1(NX1), XO(NXO)
C
C.... THIS SUBROUTINE UPDATES THE BODY COORDINATES LAMBDA - BAR
C
NEQ=NPRI*NDGREE
IMPUPD=0
IF (BETA.NE.0.) IMPUPD=1
C
J1=4
K1=1
J=J1
K=K1
DELT2=DELT/2.D0
DELTT2=DELT*DELT2
1 J2=J+1
J3=J+2
K2=K+1
K3=K+2
K4=K+3
K5=K+4
K6=K+5
K7=K+6
K8=K+7
K9=K+8
C
C.... FIND E3X, E3Y, AND E1Y
C
IF (IMPUPD.EQ.0) GO TO 2
C
DTHETX=X1(J1)-XO(J1)
DTHETY=X1(J2)-XO(J2)
DTHETZ=X1(J3)-XO(J3)
OMEGAX=V(J1)*DELT2
OMEGAZ=V(J3)*DELT2
CONX=1.D0+OMEGAX*OMEGAX
CONZ=1.D0+OMEGAZ*OMEGAZ
C
E3(1)=(OMEGAX*DTHETX+DTHETY)/CONZ
E3(2)=(OMEGAZ*DTHETY-DTHETX)/CONZ
E1(2)=(OMEGAX*DTHETY+DTHETZ)/CONX
C
GO TO 3
2 E3(1)=DELT*V(J2)+DELTT2*(V(J1)*V(J3)+A(J2))
E3(2)=-DELT*V(J1)+DELTT2*(V(J2)*V(J3)-A(J1))
E1(2)=DELT*V(J3)+DELTT2*(V(J1)*V(J2)+A(J3))
3 IF (DABS(E3(1)).LT.1.0D-20) E3(1)=0.D0
IF (DABS(E3(2)).LT.1.0D-20) E3(2)=0.D0
IF (DABS(E1(2)).LT.1.0D-20) E1(2)=0.D0
C
C.... FIND E3Z AND NORMALIZE E3
C
E31S=E3(1)*E3(1)
E32S=E3(2)*E3(2)
E3(3)=-.5D0*(E31S+E32S)
C ALSQR = E31S + E32S + ( 1.D0+E3(3) )*( 1.D0+E3(3) )
C IF ( ALSQR .LT. 0.0 ) GO TO 10
13 CONTINUE
AL=DSQRT(E31S+E32S+(1.D0+E3(3))*(1.D0+E3(3)))
E3(1)=E3(1)/AL
E3(2)=E3(2)/AL
E3(3)=(1.D0+E3(3))/AL
C
C.... COMPLETE E1
C

```

```

      E1(3)=- (E3(1)+E1(2)*E3(2))/E3(3)
C   E11SQR = 1.D0 - E1(2)*E1(2) - E1(3)*E1(3)
C   IF ( E11SQR .LT. 0.0 ) GO TO 11
      14 CONTINUE
      E1(1)=DSQRT(1.D0-E1(2)*E1(2)-E1(3)*E1(3))
C
C..... TRANSFORM TO NEW SYSTEM AND STORE IN OLD VECTOR POSITIONS)
C
      TEMP1(1)=BLAMB(K1)*E3(1)+BLAMB(K4)*E3(2)+BLAMB(K7)*E3(3)
      TEMP1(2)=BLAMB(K2)*E3(1)+BLAMB(K5)*E3(2)+BLAMB(K8)*E3(3)
      TEMP1(3)=BLAMB(K3)*E3(1)+BLAMB(K6)*E3(2)+BLAMB(K9)*E3(3)
      TEMP2(1)=BLAMB(K1)*E1(1)+BLAMB(K4)*E1(2)+BLAMB(K7)*E1(3)
      TEMP2(2)=BLAMB(K2)*E1(1)+BLAMB(K5)*E1(2)+BLAMB(K8)*E1(3)
      TEMP2(3)=BLAMB(K3)*E1(1)+BLAMB(K6)*E1(2)+BLAMB(K9)*E1(3)
      BLAMB(K1)=TEMP2(1)
      BLAMB(K2)=TEMP2(2)
      BLAMB(K3)=TEMP2(3)
      BLAMB(K4)=TEMP1(2)*TEMP2(3)-TEMP1(3)*TEMP2(2)
      BLAMB(K5)=TEMP1(3)*TEMP2(1)-TEMP1(1)*TEMP2(3)
      BLAMB(K6)=TEMP1(1)*TEMP2(2)-TEMP1(2)*TEMP2(1)
      BLAMB(K7)=TEMP1(1)
      BLAMB(K8)=TEMP1(2)
      BLAMB(K9)=TEMP1(3)
      J1=J1+6
      K1=K1+9
      J=J1
      K=K1
      IF (J1.LT.NEQ) GO TO 1
      RETURN
END

```

```

SUBROUTINE VECT (X,Y,Z,A,B,C,ABC,D1,BETA,XC,YC,ZC)
C
C   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C..... THIS ROUTINE CALCULATES THE COMPONENTS OF ELEMENT
C..... UNIT VECTORS E-1 AND E-2
C
C   DIMENSION X(2), Y(2), Z(2)
C
C   CB=DCOS(BETA)
C   SB=DSIN(BETA)
C   XC=((X(2)-X(1))*CB+(B*(Z(2)-Z(1))-C*(Y(2)-Y(1)))*SB/ABC)/D1
C   YC=((Y(2)-Y(1))*CB+(C*(X(2)-X(1))-A*(Z(2)-Z(1)))*SB/ABC)/D1
C   ZC=((Z(2)-Z(1))*CB+(A*(Y(2)-Y(1))-B*(X(2)-X(1)))*SB/ABC)/D1
C   RETURN
C   END

```

```

SUBROUTINE VECTOR (X,Y,Z,SIDE,S,A,B,C,ABC)
C
Cbrc
Cbrc CLEANED UP CODE - DJP
Cbrc
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.... THIS ROUTINE CALCULATES TRIANGULAR ELEMENT SIDE
C.... LENGTHS AND UNIT NORMAL PARAMETERS
C
      INCLUDE 'ADJUST.COM'
      DIMENSION X(3), Y(3), Z(3), SIDE(3)
C
      S=0.D0

      DO I=1,3
        J=I+1
        IF (I.EQ.3) J=1
        SIDE(I)=DSQRT((X(J)-X(I))**2+(Y(J)-Y(I))**2+(Z(J)-Z(I))**2)
        S=S+SIDE(I)
      END DO

      S=.5D0*S
      A=Y(1)*(Z(2)-Z(3))+Y(2)*(Z(3)-Z(1))+Y(3)*(Z(1)-Z(2))
      B=X(1)*(Z(3)-Z(2))+X(2)*(Z(1)-Z(3))+X(3)*(Z(2)-Z(1))
      C=X(1)*(Y(2)-Y(3))+X(2)*(Y(3)-Y(1))+X(3)*(Y(1)-Y(2))
      ABC=DSQRT(A*A+B*B+C*C)

      RETURN
      END

```

# PROGRAM WHAM3

```

C
C.....*****
C.....*****
C.....*****
C.....****
C.....
C..... STRUCTURE OF Q ARRAY ****
C.....****
C..... NAME SIZE DESCRIPTION ****
C.....****
C..... FINT 2*NDGREE*NPRI INTERNAL FORCE ARRAY ****
C..... XC NODET X COORDINATE OF NODE POINTS ****
C..... YC NODET Y COORDINATE OF NODE POINTS ****
C..... ZC NODET Z COORDINATE OF NODE POINTS ****
C..... INMESH NODMAX INTERNAL NODE NUMBER LOCATOR ****
C..... MESHIN NODET MESH NODE NUMBER LOCATOR ****
C..... LX0 MEQ OLD NODAL DISPLACEMENT ARRAY ****
C..... LX1 MEQ NEW NODAL DISPLACEMENT ARRAY ****
C..... BLAMB 9*NPRI NODAL BODY VECTOR ARRAY ****
C..... V0 N1 OLD NODAL VELOCITY ARRAY ****
C..... A0 N1 OLD NODE ACCL. ARRAY ****
C..... A1 N1 NEW NODE ACCL. ARRAY ****
C..... SMASS N1 NODAL MASS ARRAY ****
C..... FORCD N1 NEW EXTERNAL FORCE ARRAY ****
C..... FEXOLD N1 OLD EXTERNAL FORCE ARRAY ****
C..... AUX N1 DEFORMED CORRINATE ARRAY ****
C..... INDEX NELE+1 INDEX TO SRRS ARRAY ****
C..... IX NELE*7 ELEMENT ARRAY ****
C..... AL NELE ORIGINAL ELEMENT LENGTHS
C..... DICOS NELE*9 DIRECTION COSIN ARRAY ****
C..... E NUMMAT*12 MATERIAL PROPERTIES ARRAY ****
C..... NODDIS NUMDIS+1 FIXITIES ARRAY ****
C..... IPT NUMSEC NO. OF INTEGRATION PTS PER SECTIO
C..... STRS STRESS ARRAY ****
C..... STRESS 2*MAXIPT STRESS AT INTEGRATION PTS
C..... STRAIN 2*MAXIPT STRAIN AT INTEGRATION PTS
C..... YIPT MAXIPT Y COORDINATES OF INTEGRATION
C..... ZIPT MAXIPT Z COORDINATES OF INTEGRATION
C..... XLEN MAXIPT LENGTHS FOR INTEGRATION
C..... THCK NUMSEC THICKNESS FOR INTEGRATION
C..... T NPTS TIME ARRAY FOR OUTPUT ****
C..... UOUT NPRU NODAL OUTPUT ARRAY ****
C..... SOUT NPRS ELEMENT OUTPUT CODE ARRAY ****
C..... NPOUT NPIC PICTURE OUTPUT ARRAY ****
C..... GLABEL 20*NPLOT PLOTTER OUTPUT LABELS ****
C..... PSU NPTS*NPLOT OUTPUT PTS. FOR TIME HISTORY ****
C..... UU NPRU OUTPUT KINEMATIC VALUES ****
C..... SS NPRS OUTPUT FORCE VALUES ****
C..... A NPTS OUTPUT DEPENDENT VARIABLE ****
C..... NTYPE NPLOT PLOT TYPE IDENTIFIER ****
C.....****
C..... FOR SLIDING INTERFACES ONLY
C.....****
C..... NCP 10*NOPT NUMBER OF CONTACT POINTS ****
C..... NASN 3*NOPT NODE NUMBERS OF CONTACT POINT****
C..... DICOSP 3*NOPT DIRECTION OF MOTION OF PLANE
C..... ALPHA 9*NOPT UNIT NORMAL TO SLIDING PLANE ****
C..... UP 3*NOPT DISPLACEMENT OF PLANE ****
C..... UP2 3*NOPT ACCELERATION OF PLANE ****
C..... NPNO NOPT NODE NO. OF PLANES
C..... SEATK 2*NOPT STIFFNESS FOR PLANES
C..... UPOLD 3*NOPT OLD DISP OF PLANES
C..... VDAMP NOPT DAMPING COEFFICIENTS FOR PLANES
C..... UP1 3*NOPT VELOCITIES OF PLANES
C.....****
C.....*****
C.....*****
C.....*****
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
Cbrc
Cbrc COMMON INCLUDES ADDED TO ELIMINATE SOME MESSINESS - DJP
Cbrc
Cbrc INCLUDE 'ADJUST.COM'
Cbrc INCLUDE 'ARRAY.COM'

```



```

INCLUDE 'CONTRL.COM'
INCLUDE 'DYNAM.COM'
INCLUDE 'INDEX.COM'
INCLUDE 'OUTPA.COM'
INCLUDE 'SIZE.COM'
INCLUDE 'NUMINT.COM'
INCLUDE 'PRESBC.COM'
COMMON /DEVS/ LUTRM, LUTN, LUOUTP, LUPLT
COMMON /LABELS/ NAINP, NAOUT, NAPLT
INTEGER INP, OUT, PLT, LUTRM, LUTN, LUOUTP, LUPLT
CHARACTER*64 NAINP, NAOUT, NAPLT
INTEGER*2 HR, MIN, SEC, CENTS, DHR, DMIN, DSEC, DCENTS, DH, DM,
& DS, DC
CHARACTER*8 SRNAME
DIMENSION IQ(1)
DIMENSION XO(2000), X1(2000)
C DIMENSION IQT(200)
DOUBLE PRECISION DQ(1)
EQUIVALENCE (Q(1), DQ(1), IQ(1))
EQUIVALENCE (Q(1199), XO(1))
EQUIVALENCE (Q(2765), X1(1))
C EQUIVALENCE (Q(12781), IQT(1))
DIMENSION TITLE(20)
DATA REST1, REST2 /4HREST, 4HART /
C
Cbrc
Cbrc GO TO'S REPLACED WITH DO..END DO AND IF..END IF
Cbrc
MAXQ = 45000
DO LEN=1, MAXQ
  Q(LEN)=0.D0
END DO
Cbrc
Cbrc OPEN I/O DEVICES BY CALLING IODECLS - JBB
Cbrc
INP = 1
OUT = 1
PLT = 1
CALL IODECLS(INP, OUT, PLT)
C
C.... READ TITLE ****
C
READ(5,901) TITLE
WRITE (6,902) TITLE
WRITE(LUOUTP, '(/,5X, "****TIME SET TO ZERO AFTER I/O DEFS", /)')
IF(TITLE(1) .EQ. REST1 .OR. TITLE(2) .EQ. REST2) STOP
C
C.... READ IN SIZE OF PROBLEM ****
C
READ(5,903) NNODE, NPRI, NAXOR, NELE, NUMMAT, NUMDIS, MXSTEP, NDGREE,
1 DELT, NODMAX
C
C.... NET UP INDEX TO Q ARRAY ****
C
MEQ=NNODE*NDGREE
NODET=NNODE+NAXOR
IF (NODMAX.EQ.0) NODMAX=NODET
N1=NPRI*NDGREE
LFINT=1
LXC=NPRI*NDGREE+LFINT
LYC=NODET+LXC
LZC=NODET+LYC
LINMEH=NODET+LZC
NADD = 0
IF ( MOD(NODMAX,2) .NE. 0 ) NADD = 1
LMESHN = NODMAX/2 + NADD + LINMEH
NADD = 0
IF ( MOD(NODET,2) .NE. 0 ) NADD = 1
LXO = NODET/2 + NADD + LMESHN
LX1=MEQ+LXO
LBLAMB=MEQ+LX1
LVO=NPRI*9+LBLAMB
LAO=N1+LVO
LA1=N1+LAO
LSMASS=N1+LA1
LFORCD=N1+LSMASS

```

```

LFEXOD=N1+LFORCD
LAUX=N1+LFEXOD
LINDEX=MEQ+LAUX
IXADD = 0
IF ( MOD(NELE,2) .NE. 0 ) IXADD = 1
LIX = NELE/2 + 1 + IXADD + LINDEX
LAL = NELE*7 + LIX
LDICOS=NELE+LAL
LE=NELE*9+LDICOS
LNODIS=NUMMAT*12+LE
LANGLE=NUMDIS/2+LNODIS+2
LIPT=9*NUMDIS+LANGLE
WRITE (6,904) LIPT
C
NFINT = NPRI*NDGREE
NXC = NODET
NYC = NODET
NZC = NODET
NINMEH = NODMAX
NMESHN = NODET
NXO = MEQ
NX1 = MEQ
NBLAMB = NPRI
NNBLAM = 9*NPRI
NVO = N1
NAO = N1
NA1 = N1
NSMASS = N1
NFORCD = N1
NFEXOD = N1
NAUX = MEQ
NINDEX = NELE+1
NIX = NELE
NAL = NELE
NDICOS = NELE
NE = NUMMAT
NNODIS = NUMDIS+1
NANGLE = NUMDIS
C
C.... CALL READIN ****
C
CALL GETTIM(HR,MIN,SEC,CENTS)
SRNAME = ' READIN'
CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUTRM,920) SRNAME, DH, DM, DS, DC
WRITE(LUOUTP,920) SRNAME, DH, DM, DS, DC
CALL READIN(Q(LSMASS),Q(LE),Q(LXC),Q(LYC),Q(LZC),Q(LIX),Q(LXO),
1      Q(LX1),Q(LDICOS),Q(LMESHN),Q(LINMEH),Q(LNODIS),
2      Q(LANGLE))
CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUTRM,921) SRNAME, DH, DM, DS, DC
WRITE(LUOUTP,921) SRNAME, DH, DM, DS, DC
C
IF ( KONTRL(12) .NE. 0 ) THEN
MEQ2 = NNODE-NPRI
LXT = LIPT+1
LYT = MEQ2+LXT
LZT = MEQ2+LYT
LIPT = MEQ2+LZT
NXT = MEQ2
NYT = MEQ2
NZT = MEQ2
NIPT = 3*MEQ
NNODE1 = NNODE-1
DO I=NPRI,NNODE1
Q(LXT-NPRI+I) = Q(LXO+I)
Q(LYT-NPRI+I) = Q(LX1+I)
Q(LZT-NPRI+I) = Q(LDICOS+I)
END DO
END IF

```

```

NUMSEC=0
LSTRS=LIPT+1
NSTRS = 0
IF(KONTRL(9).NE.0) THEN
C
C.... READ NUMBER OF CROSS SECTIONAL GEOMETRIES
C
  READ (5,905) NUMSEC, MAXIPT
  LSTRS=LIPT+NUMSEC/2 + 1
  NSTRS = NUMSEC
C
  END IF
C
C.... ASSEMBLE ELEMENT INFORMATION
C
  NSTRS = NSTRS + NUMMAT + MAXQ/2
C
  SRNAME = ' ASSBLE'
  CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
  CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
  WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUTRM,920) SRNAME, DH, DM, DS, DC
  WRITE(LUOUTP,920) SRNAME, DH, DM, DS, DC
  CALL ASSBLE (Q(LIX),Q(LXC),Q(LYC),Q(LZC),Q(LE),Q(LSMAS),Q(LBLAMB)
  I      ,Q(LDICOS),Q(LINDEX),Q(LSTRS),Q(LIPT),Q(LAL))
  CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
  CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
  WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUTRM,921) SRNAME, DH, DM, DS, DC
  WRITE(LUOUTP,921) SRNAME, DH, DM, DS, DC
  LSTRES=IQ(2*LINDEX-1+NELE)+1+NUMMAT+LSTRS
  WRITE (6,908) LSTRES
C
  LSTRAI=LSTRES+2*MAXIPT
  LYIPT=LSTRAI+2*MAXIPT
  LZIPT=LYIPT+MAXIPT*NUMSEC
  LXLEN=LZIPT+MAXIPT*NUMSEC
  LTHCK=LXLEN+MAXIPT*NUMSEC
  LFTIME=LTHCK+2*NUMSEC
  NSTRS = LSTRES - LSTRS
  NSTRES = MAXIPT * 2
  NSTRAI = MAXIPT * 2
  NYIPT = NUMSEC
  NZIPT = NUMSEC
  NXLEN = NUMSEC
  NTHCK = NUMSEC*2
C
C.... READ PRESCRIBED BOUNDARY CONDITION DATA AND ALLOCATE SPACE
C
  READ (5,903) NFLC,MFPTS,NFNODE,NDLC,MDPTS,NDNODE,IVNODE
C
  LFFCN=LFTIME+MFPTS*NFLC
  LDTIME=LFFCN+MFPTS*NFLC
  LDFCN=LDTIME+MDPTS*NDLC
  LNODEF=LDFCN+MDPTS*NDLC
  LIDIR=LNODEF+(NFNODE+NDNODE)/2 + 1
  LNCURV=LIDIR+(NFNODE+NDNODE)/2 + 1
  LCOEF=LNCURV+(NFNODE+NDNODE)/2 + 1
  LT=LCOEF+NFNODE+NDNODE
  NFTIME = NFLC
  NFFCN = NFLC
  NDTIME = NDLC
  NDFCN = NDLC
  NNODEF = NFNODE+NDNODE
  NIDIR = NFNODE+NDNODE
  NNCURV = NFNODE+NDNODE
  NCOEF = NFNODE+NDNODE
C
C.... READ FORCE AND DISPLACEMENT DATA
C
Cbrc DOES READFD MATTER?
Cbrc  CALL READFD (Q(LFTIME),Q(LFFCN),Q(LDTIME),Q(LDFCN),Q(LNODEF),Q(LID
Cbrc  IIR),Q(LNCURV),Q(LCOEF),Q(LVO),Q(LINMEH),MFPTS,MDPTS)
Cbrc

```

```

C
C.... READ IN OUTPUT DATA ****
C
  READ (5,905) NPFREQ, NPRU, NPRS, NPIC, NANG, NPSEC
C
C.... ALLOCATE Q ARRAY FOR OUTPUT DATA ****
C
  NPTS=MXSTEP/NPFREQ+2
  NPLOT = NPRU+NPRS+3*NPSEC
  LDEF=NPTS+LT
  LUOUT=LDEF+4*NANG
  LSOUT=NPRU/2+1+LUOUT
  LNPOUT=NPRS/2+1+LSOUT
  LGLABE=2*NPIC+LNPOUT
  IF (NPIC.LT.0) LGLABE = LNPOUT+2
  LPSU = 20*NPLOT + LGLABE
  LUU=NPTS*NPLOT+LPSU
  NT = NPTS
  NDEF = NANG
  NUOUT = NPRU
  NSOUT = NPRS
  NNPOUT = NPIC
  IF (NPIC.LT.0) NNPOUT=1
  NGLABE = NPLOT
  NPSU = NPLOT
C
C  READ INFO FOR SEC. BODIES FOR CONTAC AND SET INDEX TO Q
C
  IS=1
  IF(KONTRL(6).LE.0) IS=0
  LVBRD=LUU+NPRU
  LMBRD=LVBRD+IS*45
  LABRD=LMBRD+IS*45
  LADDMAS=LABRD+IS*45
  LNCONEL=LADDMAS+120*IS
  LIPOSS=LNCONEL+64*IS
  LNELCON=LIPOSS+1*IS
  LISEC=LNELCON+1*IS
  LC=LISEC+6*KONTRL(6)
  LAREA=LC+IS*3
  LAEL=LAREA+IS*3
  LDIRCOS=LAEL+IS*15
  LIMPBRD=LDIRCOS+IS*15
  LIDEL=LIMPBRD+IS*15
  LVOLD=LIDEL+IS*15
  NUU = NPRU
  NVBRD = IS*45
  NMBRD = IS*45
  NABRD = IS*45
  NADDMAS = 120*IS
  NNCONEL = 64*IS
  NIPOSS = IS
  NNELCON = IS
  NISEC = 6*KONTRL(6)
  NC = IS*3
  NAREA = IS*3
  NAEL = IS*15
  NDIRCOS = IS*15
  NIMPBRD = IS*15
  NIDEL = IS*15
C
  SRNAME = ' READOU'
  CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
  CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
  WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUTRM,920) SRNAME, DH, DM, DS, DC
  WRITE(LUOUTP,920) SRNAME, DH, DM, DS, DC
  CALL READOU (Q(LUOUT),Q(LSOUT),Q(LNPOUT),Q(LGLABE),Q(LINMEH),
  I      Q(LIX),Q(LDEF),KONTRL(6),Q(LISEC))
  CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
  CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
  WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
  WRITE(LUTRM,921) SRNAME, DH, DM, DS, DC
  WRITE(LUOUTP,921) SRNAME, DH, DM, DS, DC

```

```

LSS=LVOLD+NI
LA=NPRS+LSS
LNTYPE=NPTS+LA
LTOTAL=NPLOT+LNTYPE
MUD2=2*MUD
IF(KONTRL(11).GE.1)
1 WRITE(6,907) LFINT,LXC,LYC,LZC,LINMEH,LMESHN,LXO,LX1,LBLAMB,
2   LVO,LAO,LA1,LSMASS,LFORCD,LFEXOD,LAUX,LINDEX,LIX,
3   LAL,LDICOS,LE,LNODIS,LANGLE,LIPT,LSTRS,LSTRES,
4   LSTRAI,LYPT,LZIPT,LXLEN,LTHCK,LFTIME,LFFCN,
5   LDTIME,LDFCN,LNODEF,LDIR,LNCURV,LCOEF,LT,LUOUT,
6   LSOUT,LNPOUT,LGLABE,LPSU,LUU,LSS,LA,LNTYPE,LTOTAL
NVOLD = NI
NSS = NPRS
NA = NPTS
NNTYPE = NPLOT
C
C ALLOCATION OF Q ARRAY FOR SLIDING INTERFACES
C
NOPT=KONTRL(4)
LNCP=LTOTAL
LNASN=5*NOPT+LNCP
LDICOP=1.5*NOPT+LNASN
LALPHA=9*NOPT+LDICOP
LUP=9*NOPT+LALPHA
LUP2=3*NOPT+LUP
LNPNO=3*NOPT+LUP2
LSEATK=NOPT/2+1+LNPNO
LUPOLD=2*NOPT+LSEATK
LVDAMP=3*NOPT+LUPOLD
LUP1=NOPT+LVDAMP
LTOTAL=3*NOPT+LUP1
IF(KONTRL(11).GE.1) WRITE(6,910) LNCP,LNASN,LDICOP,LALPHA,LUP,
1   LUP2,LNPNO,LSEATK,LUPOLD,LVDAMP,LUP1,LTOTAL
NNCP = NOPT
NNASN = 3*NOPT
NDICOP = NOPT
NALPHA = NOPT
NUP = 3*NOPT
NUP2 = 3*NOPT
NNPNO = NOPT
NSEATK = NOPT
NUPOLD = 3*NOPT
NVDAMP = NOPT
NUP1 = 3*NOPT
IF(LTOTAL.LE.MAXQ) THEN
  WRITE(6,909) LTOTAL
C
C.... CALL SOLVE FOR EXPLICIT PROBLEM ****
C
SRNAME = ' SOLVE'
CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUTRM,920) SRNAME, DH, DM, DS, DC
WRITE(LUOUTP,920) SRNAME, DH, DM, DS, DC
CALL SOLVE (Q(LNODIS),Q(LSMASS),Q(LXC),Q(LYC),Q(LZC),Q(LIX),
1   Q(LE),Q(LXO),Q(LX1),Q(LVO),Q(LAO),Q(LA1),Q(LFORCD),
2   DQ(LFINT),Q(LBLAMB),Q(LDICOS),Q(LINMEH),
3   Q(LAL),Q(LINDEX),Q(LSTRS),Q(LSTRAI),Q(LSTRES),
4   Q(LIPT),Q(LFEXOD),Q(LNCP),Q(LNASN),Q(LDICOP),
5   Q(LALPHA),Q(LUP),Q(LUP2),Q(LNPNO),Q(LSEATK),
6   Q(LUPOLD),Q(LVDAMP),Q(LUP1),Q(LVOLD))
CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUTRM,921) SRNAME, DH, DM, DS, DC
WRITE(LUOUTP,921) SRNAME, DH, DM, DS, DC
SRNAME = ' STOP'
CALL GETTIM(DHR,DMIN,DSEC,DCENTS)
CALL SUBTIM(DHR,HR,DMIN,MIN,DSEC,SEC,DCENTS,CENTS,DH,DM,DS,DC)
WRITE(LUTRM,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUOUTP,920) SRNAME, DHR, DMIN, DSEC, DCENTS
WRITE(LUTRM,920) SRNAME, DH, DM, DS, DC

```

```

WRITE(LUOUTP,920) SRNAME, DH, DM, DS, DC
STOP
WRITE (6,912) LTOTAL
END IF

```

```

STOP

```

C

```

901 FORMAT (20A4)
902 FORMAT (1H1,10X,20A4)
903 FORMAT (6I5,I6,I4,D10.3,I5)
904 FORMAT (43H TOTAL LENGTH OF Q ARRAY ENTERING READIN IS,I5)
905 FORMAT (16I5)
907 FORMAT (43H THE FOLLOWING IS THE INDEXS TO THE Q ARRAY//9H LFINT
1-,I5,9H LXC -,I5,9H LYC -,I5,9H LZC -,I5,9H LINMEH -,I5,9
2H LMESHN -,I5,9H LXO -,I5,9H LXI -,I5,9H LBLAMB -,I5,9H LVO
3 -,I5,9H LAO -,I5,9H LAI -,I5,9H LSMASS -,I5,9H LFORCD -,I
45,9H LFEXOD -,I5,9H LAUX -,I5,9H LINDEX -,I5,9H LIX -,I5,9H L
5AL -,I5,9H LDICOS -,I5,9H LE -,I5,9H LNODIS -,I5,9H LANGLE
6-,I5,9H LIPT -,I5,9H LSTRS -,I5,9H LSTRES -,I5,9H LSTRAI -,I5,9
7H LYIPT -,I5,9H LZIPT -,I5,9H LXLEN -,I5,9H LTHCK -,I5,9H LFTI
8ME -,I5,9H LFFCN -,I5,9H LDTIME -,I5,9H LDFCN -,I5,9H LNODEF -,I
95,9H LIDIR -,I5,9H LNCURV -,I5,9H LCOEF -,I5,9H LT -,I5,9H L
SUOUT -,I5,9H LSOUT -,I5,9H LNPOUT -,I5,9H LGLABE -,I5,9H LPSU
$,I5,9H LUU -,I5,9H LSS -,I5,9H LA -,I5,9H LNTYPE -,I5,9
$H LTOTAL -,I5)
908 FORMAT (49H TOTAL LENGTH OF Q ARRAY AFTER CALL TO ASSBLE IS , I5)
909 FORMAT (/10X,27HTOTAL LENGTH OF Q ARRAY IS ,I6)
910 FORMAT (44H ALLOCATION OF Q ARRAY FOR SLIDING INTERFACE/9H LNCP
1-,I5,9H LNASN -,I5,9H LDICOP -,I5,9H LALPHA -,I5,9H LUP -,I5,9
2H LUP2 -,I5,9H LNPNO -,I5,9H LSEATK -,I5,9H LUPOLD -,I5,9H LVDA
3MP -,I5,9H LUP1 -,I5,9H LTOTAL -,I5)
912 FORMAT(1H0,27HREQUIRED LENGTH OF ARRAY = ,I5,21H EXECUTION TERMIN
IATED )

```

Cbrc

Cbrc FORMATs 920 AND 921 ADDED TO PRINT TIMING INFORMATION

Cbrc

```

920 FORMAT(//,5X,'*****TIME OF CALL TO SR ',A8,' ',I2,' ',I2,' ',I2,
+ ' ',I2,/)
921 FORMAT(//,5X,'*****TIME OF RETURN FROM SR ',A8,' ',I2,' ',I2,
+ ' ',I2,' ',I2,/)
END

```

```

COMMON /ADJUST/ NFINT,NXC,NYC,NZC,NINMEH,NMESH,NXO,NX1,NBLAMB,
1      NVO,NAO,NA1,NSMASS,NFORCD,NFEXOD,NAUX,NINDEX,
2      NIX,NAL,NDICOS,NE,NNODIS,NANGLE,NXT,NTY,NZT,
3      NIPT,NSTRS,NSTRES,NSTRAI,NYIPT,NZIPT,NXLEN,
4      NTHCK,NFTIME,NFFCN,NDTIME,NDFCN,NNODEF,NIDIR,
5      NNCURV,NCOEF,NT,NDEF,NUOUT,NSOUT,NNPOUT,
6      NGLABE,NPSU,NUU,NVBRD,NMBRD,NABRD,NADDMAS,
7      NNCONEL,NIPOSS,NNELCON,NISEC,NC,NAREA,NAEL,
8      NDIRCOS,NIMPBRD,NIDEL,NVOLD,NSS,NA,NNTYPE,
9      NNCP,NNASN,NDICOP,NALPHA,NUP,NUP2,NNPNO,
A      NSEATK,NUPOLD,NVDAMP,NUP1,NNBLAM

```

```

COMMON /ARRAY/ MAXQ, DUM1, Q(45000)

```

```

COMMON /CONTRL/ KONTRL(16)

```

```

COMMON /DYNAM/ DELT,TIMENP,MXSTEP,NTSTEP,TIME

```

```

COMMON /INDEX/ LFINT,LXC,LYC,LZC,LINMEH,LMESH,NLXO,LX1,LBLAMB,LVO,
1LAO,LA1,LSMASS,LFORCD,LFEXOD,LAUX,LINDEX,LIX,LAL,LDICOS,LE,LNODIS,
2LANGLE,LIPT,LSTRS,LSTRES,LSTRAI,LYIPT,LZIPT,LXLEN,LTHCK,LFTIME,LFF
3CN,LDTIME,LDFCN,LNODEF,LIDIR,LNCURV,LCOEF,LT,LUOUT,LSOUT,LNPOUT,LG
4LABE,LPSU,LUU,LSS,LA,LNTYPE,LNCP,LNASN,LDICOP,LALPHA,LUP,LUP2,LNPN
5O,LSEATK,LUPOLD,LVDAMP,LUP1,LVBRD,LMBRD,LABRD,LADDMAS,LNCONEL,
. LIPOSS,LNELCON,LISEC,LC,LAREA,LAEL,LDIRCOS,LIMPBRD,LIDEL,LVOLD,
. LTOTAL,LDEF

```

```

COMMON /MATRIX/ NAM,NB,NR

```

```

COMMON /NUMINT/ NUMSEC,MAXIPT

```

```

COMMON /OUTPA/ NPRU, NPRS, NPFREQ, NPIC, NPLOT, NPTS, NANG, NPSEC

```

```

COMMON /PRESBC/ NFLC,MFPTS,NFNODE,NDLC,MDPTS,NDNODE,IVNODE

```

```

COMMON /SIZE/ NNODE,NPRI,NAXOR,NELE,NUMMAT,NUMDIS,NPDIS,NDGREE,NOD
1      MAX,MUD

```

## **APPENDIX B**

### **VISUAL BASIC CODE FOR THE INPUT FILE INTERFACE**



## HSM User Interface Instructions

### Installing HSM

An installation program is provided to install HSM. To begin the installation, insert HSM disk #1 into floppy drive A:, select Run from the Windows File Menu, and then type in A:\SETUP.EXE. Click on OK and follow the instructions given on the screen. You will be prompted when to insert disk #2 into drive A. When the installation process is complete the HSM group and icon will be created for you.

### Running HSM

After the HSM software is installed, follow these steps to run HSM.

1. Double click on the HSM icon and the program will start.
2. Select an input file using the **Open** option on the **File** menu.
3. Make any editing changes required to the data from the input file.  
**Note:** Some data cannot be changed without also changing the source code for the Fortran HSM program and then recompiling.
4. Save the revised input file using the **Save** option on the **File** menu.
5. Select **Solve** from the main menu to run the Fortran HSM program. A blank DOS screen will appear while this program is running. If a slow processor is being used, this program will require several minutes to complete execution.
6. Select **Plot** from the main menu to display the plots from the output file.
7. Exit HSM by selecting **Exit** on the **File** menu.

Option Explicit

### Sub Form\_Load ()

```
'This routine initialize variables and grids
' Subroutines called:  Materials
'                      NodeData
'                      Element
'                      Displace
'                      Motion
'                      Stress
'                      Picture
'                      FunctionSpec
'                      Plane
'                      Spine
```

```
NUMMAT = 1
tabHSMInput.CurrTab = 0
```

```
'setup grids for input file
Call Materials
Call NodeData
Call Element
Call Displace
Call Motion
Call Stress
Call Picture
Call FunctionSpec
Call Plane
Call Spine
```

End Sub

### Sub mnuFileExit\_Click ()

End

End Sub

### Sub mnuFileOpen\_Click ()

```
Call ReadInputFile
```

End Sub

### Sub mnuFileSave\_Click ()

```
Call SaveInputFile
```

End Sub

### Sub mnuPlot\_Click ()

```
Call ReadOutFile
```

End Sub

### Sub mnuSolve\_Click ()

```
'This routine copies the current input file to hsm.inp and shells to DOS
'to run the Fortran executable HSM program.
```

```
Dim x 'return value from shell
```

```
If Infile <> "" Then
    FileCopy Infile, App.Path & "\hsm.inp"
```

```
ChDir "c:\hsm"  
x = Shell(App.Path & "\brchsm.exe", 2)  
While GetModuleUsage(x) > 0  
    DoEvents  
Wend  
End If  
  
End Sub
```

C:\HSMINPUT\PLOT.FRM 11/30/95 11:32:26 AM

Option Explicit

Sub cmdClose\_Click ()

frmPlot.Hide

End Sub

Option Explicit

**Sub Displace ()***'label columns for displacement nodes*

```
frmHSMInput.tblDisplace.ColumnNames(1) = "NODE"
frmHSMInput.tblDisplace.ColumnNames(2) = "TRANS. X DF"
frmHSMInput.tblDisplace.ColumnNames(3) = "TRANS. Y DF"
frmHSMInput.tblDisplace.ColumnNames(4) = "TRANS. Z DF"
frmHSMInput.tblDisplace.ColumnNames(5) = "ROT. ABOUT X"
frmHSMInput.tblDisplace.ColumnNames(6) = "ROT. ABOUT Y"
frmHSMInput.tblDisplace.ColumnNames(7) = "ROT. ABOUT Z"
```

*'size columns for displacement nodes*

```
frmHSMInput.tblDisplace.ColumnSize(1) = 4
frmHSMInput.tblDisplace.ColumnSize(2) = 1
frmHSMInput.tblDisplace.ColumnSize(3) = 1
frmHSMInput.tblDisplace.ColumnSize(4) = 1
frmHSMInput.tblDisplace.ColumnSize(5) = 1
frmHSMInput.tblDisplace.ColumnSize(6) = 1
frmHSMInput.tblDisplace.ColumnSize(7) = 1
frmHSMInput.tblDisplace.ColumnWidth(1) = 6
frmHSMInput.tblDisplace.ColumnWidth(2) = 16
frmHSMInput.tblDisplace.ColumnWidth(3) = 16
frmHSMInput.tblDisplace.ColumnWidth(4) = 16
frmHSMInput.tblDisplace.ColumnWidth(5) = 16
frmHSMInput.tblDisplace.ColumnWidth(6) = 16
frmHSMInput.tblDisplace.ColumnWidth(7) = 16
```

**End Sub****Sub Element ()***'label columns for elements*

```
frmHSMInput.tblElements.ColumnNames(1) = "No."
frmHSMInput.tblElements.ColumnNames(2) = "N1"
frmHSMInput.tblElements.ColumnNames(3) = "N2"
frmHSMInput.tblElements.ColumnNames(4) = "N3"
frmHSMInput.tblElements.ColumnNames(5) = "N4"
frmHSMInput.tblElements.ColumnNames(6) = "N5"
frmHSMInput.tblElements.ColumnNames(7) = "N6"
frmHSMInput.tblElements.ColumnNames(8) = "N7"
frmHSMInput.tblElements.ColumnNames(9) = "COOR"
frmHSMInput.tblElements.ColumnNames(10) = "MTYP"
frmHSMInput.tblElements.ColumnNames(11) = "ETYP"
frmHSMInput.tblElements.ColumnNames(12) = "N1"
frmHSMInput.tblElements.ColumnNames(13) = "N2"
frmHSMInput.tblElements.ColumnNames(14) = "N3"
frmHSMInput.tblElements.ColumnNames(15) = "N4"
frmHSMInput.tblElements.ColumnNames(16) = "COOR"
```

*'size columns for elements*

```
frmHSMInput.tblElements.ColumnSize(1) = 5
frmHSMInput.tblElements.ColumnSize(2) = 5
frmHSMInput.tblElements.ColumnSize(3) = 5
frmHSMInput.tblElements.ColumnSize(4) = 5
frmHSMInput.tblElements.ColumnSize(5) = 5
frmHSMInput.tblElements.ColumnSize(6) = 5
frmHSMInput.tblElements.ColumnSize(7) = 5
frmHSMInput.tblElements.ColumnSize(8) = 5
frmHSMInput.tblElements.ColumnSize(9) = 5
frmHSMInput.tblElements.ColumnSize(10) = 5
frmHSMInput.tblElements.ColumnSize(11) = 5
frmHSMInput.tblElements.ColumnSize(12) = 5
frmHSMInput.tblElements.ColumnSize(13) = 5
frmHSMInput.tblElements.ColumnSize(14) = 5
frmHSMInput.tblElements.ColumnSize(15) = 5
frmHSMInput.tblElements.ColumnSize(16) = 5
frmHSMInput.tblElements.ColumnWidth(1) = 8
frmHSMInput.tblElements.ColumnWidth(2) = 12
frmHSMInput.tblElements.ColumnWidth(3) = 12
```

```

frmHSMInput.tblElements.ColumnWidth(4) = 12
frmHSMInput.tblElements.ColumnWidth(5) = 12
frmHSMInput.tblElements.ColumnWidth(6) = 1
frmHSMInput.tblElements.ColumnWidth(7) = 1
frmHSMInput.tblElements.ColumnWidth(8) = 1
frmHSMInput.tblElements.ColumnWidth(9) = 12
frmHSMInput.tblElements.ColumnWidth(10) = 12
frmHSMInput.tblElements.ColumnWidth(11) = 12
frmHSMInput.tblElements.ColumnWidth(12) = 1
frmHSMInput.tblElements.ColumnWidth(13) = 12
frmHSMInput.tblElements.ColumnWidth(14) = 1
frmHSMInput.tblElements.ColumnWidth(15) = 1
frmHSMInput.tblElements.ColumnWidth(16) = 12

```

End Sub

### Sub FunctionSpec ()

```

'label columns for functions specs
frmHSMInput.tblFuncSpec.ColumnName(1) = " TIME t"
frmHSMInput.tblFuncSpec.ColumnName(2) = " f(t)"
frmHSMInput.tblFuncSpec.ColumnName(3) = " f'(t)"

'size columns for materials properties
frmHSMInput.tblFuncSpec.ColumnSize(1) = 10
frmHSMInput.tblFuncSpec.ColumnSize(2) = 10
frmHSMInput.tblFuncSpec.ColumnSize(3) = 10
frmHSMInput.tblFuncSpec.ColumnWidth(1) = 15
frmHSMInput.tblFuncSpec.ColumnWidth(2) = 15
frmHSMInput.tblFuncSpec.ColumnWidth(3) = 15

```

End Sub

### Sub Materials ()

```

'label columns for materials properties
frmHSMInput.tblMatProp.ColumnName(1) = "MTYP"
frmHSMInput.tblMatProp.ColumnName(2) = "LTP"
frmHSMInput.tblMatProp.ColumnName(3) = "E1"
frmHSMInput.tblMatProp.ColumnName(4) = "E2"
frmHSMInput.tblMatProp.ColumnName(5) = "E3"
frmHSMInput.tblMatProp.ColumnName(6) = "E4"
frmHSMInput.tblMatProp.ColumnName(7) = "E5"
frmHSMInput.tblMatProp.ColumnName(8) = "E6"
frmHSMInput.tblMatProp.ColumnName(9) = "E7"
frmHSMInput.tblMatProp.ColumnName(10) = "E8"
frmHSMInput.tblMatProp.ColumnName(11) = "E9"
frmHSMInput.tblMatProp.ColumnName(12) = "E10"
frmHSMInput.tblMatProp.ColumnName(13) = "E11"
frmHSMInput.tblMatProp.ColumnName(14) = "E12"

'size columns for materials properties
frmHSMInput.tblMatProp.ColumnSize(1) = 5
frmHSMInput.tblMatProp.ColumnSize(2) = 5
frmHSMInput.tblMatProp.ColumnSize(3) = 10
frmHSMInput.tblMatProp.ColumnSize(4) = 10
frmHSMInput.tblMatProp.ColumnSize(5) = 10
frmHSMInput.tblMatProp.ColumnSize(6) = 10
frmHSMInput.tblMatProp.ColumnSize(7) = 10
frmHSMInput.tblMatProp.ColumnSize(8) = 10
frmHSMInput.tblMatProp.ColumnSize(9) = 10
frmHSMInput.tblMatProp.ColumnSize(10) = 10
frmHSMInput.tblMatProp.ColumnSize(11) = 10
frmHSMInput.tblMatProp.ColumnSize(12) = 10
frmHSMInput.tblMatProp.ColumnSize(13) = 10
frmHSMInput.tblMatProp.ColumnSize(14) = 10
frmHSMInput.tblMatProp.ColumnWidth(1) = 8
frmHSMInput.tblMatProp.ColumnWidth(2) = 8
frmHSMInput.tblMatProp.ColumnWidth(3) = 12
frmHSMInput.tblMatProp.ColumnWidth(4) = 12
frmHSMInput.tblMatProp.ColumnWidth(5) = 12
frmHSMInput.tblMatProp.ColumnWidth(6) = 12
frmHSMInput.tblMatProp.ColumnWidth(7) = 12
frmHSMInput.tblMatProp.ColumnWidth(8) = 12
frmHSMInput.tblMatProp.ColumnWidth(9) = 12

```

```

frmHSMInput.tblMatProp.ColumnWidth(10) = 12
frmHSMInput.tblMatProp.ColumnWidth(11) = 12
frmHSMInput.tblMatProp.ColumnWidth(12) = 12
frmHSMInput.tblMatProp.ColumnWidth(13) = 12
frmHSMInput.tblMatProp.ColumnWidth(14) = 12

```

**'setup edit masks for materials properties**

```

frmHSMInput.tblMatProp.EditMask(3) = "Scientific"
frmHSMInput.tblMatProp.EditMask(4) = "Scientific"
frmHSMInput.tblMatProp.EditMask(5) = "Scientific"
frmHSMInput.tblMatProp.EditMask(6) = "Scientific"
frmHSMInput.tblMatProp.EditMask(7) = "Scientific"
frmHSMInput.tblMatProp.EditMask(8) = "Scientific"
frmHSMInput.tblMatProp.EditMask(9) = "Scientific"
frmHSMInput.tblMatProp.EditMask(10) = "Scientific"
frmHSMInput.tblMatProp.EditMask(11) = "Scientific"
frmHSMInput.tblMatProp.EditMask(12) = "Scientific"
frmHSMInput.tblMatProp.EditMask(13) = "Scientific"
frmHSMInput.tblMatProp.EditMask(14) = "Scientific"

```

**End Sub**

**Sub Motion ()**

**'label columns for motion lines**

```

frmHSMInput.tblMotion.ColumnName(1) = "UOUT"
frmHSMInput.tblMotion.ColumnName(2) = "J"
frmHSMInput.tblMotion.ColumnName(3) = "K"
frmHSMInput.tblMotion.ColumnName(4) = "L"
frmHSMInput.tblMotion.ColumnName(5) = "LABEL"

```

**'size columns for motion lines**

```

frmHSMInput.tblMotion.ColumnSize(1) = 7
frmHSMInput.tblMotion.ColumnSize(2) = 1
frmHSMInput.tblMotion.ColumnSize(3) = 1
frmHSMInput.tblMotion.ColumnSize(4) = 1
frmHSMInput.tblMotion.ColumnSize(5) = 40
frmHSMInput.tblMotion.ColumnWidth(1) = 10
frmHSMInput.tblMotion.ColumnWidth(2) = 5
frmHSMInput.tblMotion.ColumnWidth(3) = 5
frmHSMInput.tblMotion.ColumnWidth(4) = 5
frmHSMInput.tblMotion.ColumnWidth(5) = 40

```

**End Sub**

**Sub NodeData ()**

**'label columns for nodal data**

```

frmHSMInput.tblNodalData.ColumnName(1) = "Node"
frmHSMInput.tblNodalData.ColumnName(2) = "I"
frmHSMInput.tblNodalData.ColumnName(3) = "I"
frmHSMInput.tblNodalData.ColumnName(4) = "N"
frmHSMInput.tblNodalData.ColumnName(5) = "X-ORD"
frmHSMInput.tblNodalData.ColumnName(6) = "Y-ORD"
frmHSMInput.tblNodalData.ColumnName(7) = "Z-ORD"
frmHSMInput.tblNodalData.ColumnName(8) = "TMASS(1)"
frmHSMInput.tblNodalData.ColumnName(9) = "TMASS(2)"
frmHSMInput.tblNodalData.ColumnName(10) = "TMASS(3)"
frmHSMInput.tblNodalData.ColumnName(11) = "TMASS(4)"

```

**'size columns for nodal data**

```

frmHSMInput.tblNodalData.ColumnSize(1) = 5
frmHSMInput.tblNodalData.ColumnSize(2) = 1
frmHSMInput.tblNodalData.ColumnSize(3) = 1
frmHSMInput.tblNodalData.ColumnSize(4) = 1
frmHSMInput.tblNodalData.ColumnSize(5) = 10
frmHSMInput.tblNodalData.ColumnSize(6) = 10
frmHSMInput.tblNodalData.ColumnSize(7) = 10
frmHSMInput.tblNodalData.ColumnSize(8) = 10
frmHSMInput.tblNodalData.ColumnSize(9) = 10
frmHSMInput.tblNodalData.ColumnSize(10) = 10
frmHSMInput.tblNodalData.ColumnSize(11) = 10
frmHSMInput.tblNodalData.ColumnWidth(1) = 8
frmHSMInput.tblNodalData.ColumnWidth(2) = 5
frmHSMInput.tblNodalData.ColumnWidth(3) = 5

```

```

frmHSMInput.tblNodalData.ColumnWidth(4) = 5
frmHSMInput.tblNodalData.ColumnWidth(5) = 12
frmHSMInput.tblNodalData.ColumnWidth(6) = 12
frmHSMInput.tblNodalData.ColumnWidth(7) = 12
frmHSMInput.tblNodalData.ColumnWidth(8) = 12
frmHSMInput.tblNodalData.ColumnWidth(9) = 12
frmHSMInput.tblNodalData.ColumnWidth(10) = 12
frmHSMInput.tblNodalData.ColumnWidth(11) = 12

```

```

' setup edit masks for nodal data
frmHSMInput.tblNodalData.EditMask(8) = "Scientific"
frmHSMInput.tblNodalData.EditMask(9) = "Scientific"
frmHSMInput.tblNodalData.EditMask(10) = "Scientific"
frmHSMInput.tblNodalData.EditMask(11) = "Scientific"

```

**End Sub**

**Sub Picture ()**

```

' label columns for picture lines
frmHSMInput.tblPicLines.ColumnName(1) = "NPOUT"
frmHSMInput.tblPicLines.ColumnName(2) = "KON"

```

```

' size columns for picture lines
frmHSMInput.tblPicLines.ColumnSize(1) = 10
frmHSMInput.tblPicLines.ColumnSize(2) = 10
frmHSMInput.tblPicLines.ColumnWidth(1) = 15
frmHSMInput.tblPicLines.ColumnWidth(2) = 15

```

**End Sub**

**Sub Plane ()**

```

' label columns for plane lines
frmHSMInput.tblPlaneLines.ColumnName(1) = " NODE "
frmHSMInput.tblPlaneLines.ColumnName(2) = " # Contact "
frmHSMInput.tblPlaneLines.ColumnName(3) = " DIR. COS x "
frmHSMInput.tblPlaneLines.ColumnName(4) = " DIR. COS y "
frmHSMInput.tblPlaneLines.ColumnName(5) = " DIR. COS z "
frmHSMInput.tblPlaneLines.ColumnName(6) = " LINEAR "
frmHSMInput.tblPlaneLines.ColumnName(7) = " CUBIC "
frmHSMInput.tblPlaneLines.ColumnName(8) = " VISCOUS "
frmHSMInput.tblPlaneLines.ColumnName(9) = " First Node "
frmHSMInput.tblPlaneLines.ColumnName(10) = " Last Node "
frmHSMInput.tblPlaneLines.ColumnName(11) = " "
frmHSMInput.tblPlaneLines.ColumnName(12) = " "
frmHSMInput.tblPlaneLines.ColumnName(13) = " "
frmHSMInput.tblPlaneLines.ColumnName(14) = " "
frmHSMInput.tblPlaneLines.ColumnName(15) = " "
frmHSMInput.tblPlaneLines.ColumnName(16) = " "
frmHSMInput.tblPlaneLines.ColumnName(17) = " "
frmHSMInput.tblPlaneLines.ColumnName(18) = " "
frmHSMInput.tblPlaneLines.ColumnName(19) = " x1 "
frmHSMInput.tblPlaneLines.ColumnName(20) = " y1 "
frmHSMInput.tblPlaneLines.ColumnName(21) = " z1 "
frmHSMInput.tblPlaneLines.ColumnName(22) = " x2 "
frmHSMInput.tblPlaneLines.ColumnName(23) = " y2 "
frmHSMInput.tblPlaneLines.ColumnName(24) = " z2 "
frmHSMInput.tblPlaneLines.ColumnName(25) = " x3 "
frmHSMInput.tblPlaneLines.ColumnName(26) = " y3 "
frmHSMInput.tblPlaneLines.ColumnName(27) = " z3 "

```

```

' size columns for plane lines
frmHSMInput.tblPlaneLines.ColumnSize(1) = 5
frmHSMInput.tblPlaneLines.ColumnSize(2) = 5
frmHSMInput.tblPlaneLines.ColumnSize(3) = 10
frmHSMInput.tblPlaneLines.ColumnSize(4) = 10
frmHSMInput.tblPlaneLines.ColumnSize(5) = 10
frmHSMInput.tblPlaneLines.ColumnSize(6) = 10
frmHSMInput.tblPlaneLines.ColumnSize(7) = 10
frmHSMInput.tblPlaneLines.ColumnSize(8) = 10
frmHSMInput.tblPlaneLines.ColumnSize(9) = 10
frmHSMInput.tblPlaneLines.ColumnSize(10) = 10
frmHSMInput.tblPlaneLines.ColumnSize(11) = 1
frmHSMInput.tblPlaneLines.ColumnSize(12) = 1

```



```

frmHSMInput.tblPlaneLines.ColumnSize(13) = 1
frmHSMInput.tblPlaneLines.ColumnSize(14) = 1
frmHSMInput.tblPlaneLines.ColumnSize(15) = 1
frmHSMInput.tblPlaneLines.ColumnSize(16) = 1
frmHSMInput.tblPlaneLines.ColumnSize(17) = 1
frmHSMInput.tblPlaneLines.ColumnSize(18) = 1
frmHSMInput.tblPlaneLines.ColumnSize(19) = 10
frmHSMInput.tblPlaneLines.ColumnSize(20) = 10
frmHSMInput.tblPlaneLines.ColumnSize(21) = 10
frmHSMInput.tblPlaneLines.ColumnSize(22) = 10
frmHSMInput.tblPlaneLines.ColumnSize(23) = 10
frmHSMInput.tblPlaneLines.ColumnSize(24) = 10
frmHSMInput.tblPlaneLines.ColumnSize(25) = 10
frmHSMInput.tblPlaneLines.ColumnSize(26) = 10
frmHSMInput.tblPlaneLines.ColumnSize(27) = 10
frmHSMInput.tblPlaneLines.ColumnWidth(1) = 8
frmHSMInput.tblPlaneLines.ColumnWidth(2) = 5
frmHSMInput.tblPlaneLines.ColumnWidth(3) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(4) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(5) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(6) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(7) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(8) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(9) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(10) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(11) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(12) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(13) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(14) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(15) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(16) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(17) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(18) = 1
frmHSMInput.tblPlaneLines.ColumnWidth(19) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(20) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(21) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(22) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(23) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(24) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(25) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(26) = 12
frmHSMInput.tblPlaneLines.ColumnWidth(27) = 12

```

**End Sub**

**Sub Spine ()**

```

'label columns for spine
frmHSMInput.tblSpine.ColumnName(1) = "   NODE   "
frmHSMInput.tblSpine.ColumnName(2) = "   AXIAL COMP"
frmHSMInput.tblSpine.ColumnName(3) = "   LAT. BENDING"
frmHSMInput.tblSpine.ColumnName(4) = "ANT/POST BENDING"

'size columns for spine
frmHSMInput.tblSpine.ColumnSize(1) = 5
frmHSMInput.tblSpine.ColumnSize(2) = 5
frmHSMInput.tblSpine.ColumnSize(3) = 5
frmHSMInput.tblSpine.ColumnSize(4) = 5
frmHSMInput.tblSpine.ColumnWidth(1) = 10
frmHSMInput.tblSpine.ColumnWidth(2) = 20
frmHSMInput.tblSpine.ColumnWidth(3) = 20
frmHSMInput.tblSpine.ColumnWidth(4) = 20

```

**End Sub**

**Sub Stress ()**

```

'label columns for stress lines
frmHSMInput.tblStress.ColumnName(1) = "SOUT"
frmHSMInput.tblStress.ColumnName(2) = "I1"
frmHSMInput.tblStress.ColumnName(3) = "I2"
frmHSMInput.tblStress.ColumnName(4) = "I3"
frmHSMInput.tblStress.ColumnName(5) = "M"
frmHSMInput.tblStress.ColumnName(6) = "N"
frmHSMInput.tblStress.ColumnName(7) = "LABEL"

```

```
'size columns for stress lines
frmHSMInput.tblStress.ColumnSize(1) = 5
frmHSMInput.tblStress.ColumnSize(2) = 1
frmHSMInput.tblStress.ColumnSize(3) = 1
frmHSMInput.tblStress.ColumnSize(4) = 1
frmHSMInput.tblStress.ColumnSize(5) = 1
frmHSMInput.tblStress.ColumnSize(6) = 1
frmHSMInput.tblStress.ColumnSize(7) = 40
frmHSMInput.tblStress.ColumnWidth(1) = 10
frmHSMInput.tblStress.ColumnWidth(2) = 5
frmHSMInput.tblStress.ColumnWidth(3) = 5
frmHSMInput.tblStress.ColumnWidth(4) = 5
frmHSMInput.tblStress.ColumnWidth(5) = 5
frmHSMInput.tblStress.ColumnWidth(6) = 5
frmHSMInput.tblStress.ColumnWidth(7) = 40
```

**End Sub**

Option Explicit

```

Global Infile As String      ' name of input file
Global Outfile As String    ' name of output file

' parameter variables
Global NNODE As Integer     ' number of nodes in the model
Global NPRI As Integer      ' number of primary nodes in the model
Global NAXOR As Integer     ' number of axis orientation nodes in the model
Global NELE As Long         ' number of elements in the model
Global NUMMAT As Long       ' number of different element section and material types
Global NUMDIS As Integer    ' number of nodes at which any displacement components are specified
Global MXSTEP As Long       ' number of time steps to be taken
Global NDGREE As Integer    ' number of degrees of freedom per node
Global DELT As Double       ' time increment
Global NODMAX As Integer    ' largest node number in model

' input variables
Global NLOAD As Integer     ' load
Global NIC As Integer       ' displacement

' output control variables
Global NPFREQ As Integer    ' frequency of output
Global NPRU As Integer      ' number of motion output lines
Global NPRS As Integer      ' number of stress output lines
Global NPIC As Integer      ' number of complete output pictures
Global NANG As Integer      ' used when ELMPLT is desired in output
Global NUMSEC As Integer    ' sets of cross-sectional geometries
Global NPTS As Integer      ' number of points where the motion functions value and derivative are
»» specified
Global F1 As Double         ' integration constant for first integration
Global F2 As Double         ' integration constant for second integration
Global JSTART As Integer    ' inferior disc element for bottommost vertebral level considered
Global JEND As Integer      ' superior disc element for uppermost vertebral level considered
Global NB As Integer        ' number of vertebrae
Global PLTDICD As Integer   '
Global PLTLAB As Integer    '
Global IOPTPLT As Integer   '
Global TOTALX As Integer    ' width of plot (x,z) view
Global TOTALY As Integer    ' width of plot (y,z) view
Global TOTALZ As Integer    ' height of plot
Global XNEG As Integer      ' distance of the origin from left side of plot
Global YNEG As Integer      ' distance of the origin from left side of plot
Global ZNEG As Integer      ' distance of the origin from bottom of plot

' array variables
Global KONTRL(16) As Integer ' program control lines
Global NCP(2) As String     ' primary nodes associated with a plane
Global TITLE(3) As String   ' title for output
Global aryMAT() As String    ' material-section property lines
Global aryNOD() As String    ' nodal data lines
Global aryELE() As String    ' element data lines
Global aryDIS() As String    ' displacement lines
Global aryIPT() As String    ' cross section
Global aryMOT() As String    ' motion output lines
Global arySTR() As String    ' stress output lines
Global aryPIC() As String    ' output picture lines
Global aryFSL() As String    ' function specification lines
Global aryNOP() As String    ' plane identification lines
Global arySPN() As String    ' spinal injury function lines
Global aryVEC() As String    ' X-Bar, Y-Bar Vectors

```

Declare Function GetModuleUsage Lib "Kernel" (ByVal hModule As Integer) As Integer

## Sub CrossSection (buffer)

'This routine extracts fields from cross-sectional geometries lines

Dim j As Integer      'counter

```

For j = 1 To NUMSEC
    aryIPT(j) = Mid$(buffer, (j - 1) * 5 + 1, 5)
Next j

```

End Sub

Sub Displacement (buffer, i)

*'This routine extracts fields from prescribed displacement lines*

```

aryDIS(i, 1) = Mid$(buffer, 1, 4)      ' node number
aryDIS(i, 2) = Mid$(buffer, 5, 1)      ' translational x
aryDIS(i, 3) = Mid$(buffer, 6, 1)      ' translational y
aryDIS(i, 4) = Mid$(buffer, 7, 1)      ' translational z
aryDIS(i, 5) = Mid$(buffer, 8, 1)      ' rotation x
aryDIS(i, 6) = Mid$(buffer, 9, 1)      ' rotation y
aryDIS(i, 7) = Mid$(buffer, 10, 1)     ' rotation z

```

End Sub

Sub Elements (buffer, i)

*'This routine extracts fields from element data lines*

```

aryELE(i, 1) = Mid$(buffer, 1, 5)      ' node number
aryELE(i, 2) = Mid$(buffer, 6, 5)      ' Node I
aryELE(i, 3) = Mid$(buffer, 11, 5)     ' Node J
aryELE(i, 4) = Mid$(buffer, 16, 5)     ' primary node for I
aryELE(i, 5) = Mid$(buffer, 21, 5)     ' primary node for J
aryELE(i, 6) = Mid$(buffer, 26, 5)     ' not used
aryELE(i, 7) = Mid$(buffer, 31, 5)     ' not used
aryELE(i, 8) = Mid$(buffer, 36, 5)     ' not used
aryELE(i, 9) = Mid$(buffer, 41, 5)     ' Node K
aryELE(i, 10) = Mid$(buffer, 46, 5)     ' material type
aryELE(i, 11) = Mid$(buffer, 51, 5)     ' element type
aryELE(i, 12) = Mid$(buffer, 56, 5)     ' not used
aryELE(i, 13) = Mid$(buffer, 61, 5)     ' number of sliding node pairs
aryELE(i, 14) = Mid$(buffer, 66, 5)     ' not used
aryELE(i, 15) = Mid$(buffer, 71, 5)     ' not used
aryELE(i, 16) = Mid$(buffer, 76, 5)     ' level

```

End Sub

Function FormatNum (num, w) As String

*'This function converts a number to a string and pads  
'it with blanks on the left to achieve width w.  
' num = number to convert  
' w = desired width*

```

Dim str1 As String

str1 = Mid$(Str$(num), 2)
While (Len(str1) < w)
    str1 = " " & str1
Wend
FormatNum = str1

```

End Function

Sub ICIF (buffer, i)

*'This routine extracts fields from ICIF data lines*

```

aryFSL(i, 1) = Mid$(buffer, 1, 10)     ' Time t
aryFSL(i, 2) = Mid$(buffer, 11, 10)    ' motion function value at time t
aryFSL(i, 3) = Mid$(buffer, 21, 10)    ' derivative of function at time t

```

End Sub

### Sub MaterialPropertyLine1 (buffer, i)

*'This routine extracts fields from material property lines (type 1)*

```
aryMAT(i, 1) = Mid$(buffer, 1, 5)      ' MTYP
aryMAT(i, 2) = Mid$(buffer, 6, 5)      ' LTYP
```

End Sub

### Sub MaterialPropertyLine2 (buffer, i)

*'This routine extracts fields from material property lines (type 1)*

```
aryMAT(i, 3) = Mid$(buffer, 1, 10)     ' E(1,MTYP)
aryMAT(i, 4) = Mid$(buffer, 11, 10)    ' E(2,MTYP)
aryMAT(i, 5) = Mid$(buffer, 21, 10)    ' E(3,MTYP)
aryMAT(i, 6) = Mid$(buffer, 31, 10)    ' E(4,MTYP)
aryMAT(i, 7) = Mid$(buffer, 41, 10)    ' E(5,MTYP)
aryMAT(i, 8) = Mid$(buffer, 51, 10)    ' E(6,MTYP)
```

End Sub

### Sub MaterialPropertyLine3 (buffer, i)

*'This routine extracts fields from material property lines (type 1)*

```
aryMAT(i, 9) = Mid$(buffer, 1, 10)     ' E(7,MTYP)
aryMAT(i, 10) = Mid$(buffer, 11, 10)    ' E(8,MTYP)
aryMAT(i, 11) = Mid$(buffer, 21, 10)    ' E(9,MTYP)
aryMAT(i, 12) = Mid$(buffer, 31, 10)    ' E(10,MTYP)
aryMAT(i, 13) = Mid$(buffer, 41, 10)    ' E(11,MTYP)
aryMAT(i, 14) = Mid$(buffer, 51, 10)    ' E(12,MTYP)
```

End Sub

### Sub MotionLines (buffer, i)

*'This routine extracts fields from motion output lines*

```
aryMOT(i, 1) = Mid$(buffer, 1, 7)      ' node number
aryMOT(i, 2) = Mid$(buffer, 8, 1)      ' component number of kinematic variable to be output
aryMOT(i, 3) = Mid$(buffer, 9, 1)      ' displacement, velocity, or acceleration (0,1,2)
aryMOT(i, 4) = Mid$(buffer, 10, 1)     ' plot control (0-4)
aryMOT(i, 5) = Mid$(buffer, 21, 40)    ' label
```

End Sub

### Sub NodalData (buffer, i)

*'This routine extracts fields from nodal data lines*

```
aryNOD(i, 1) = Mid$(buffer, 1, 5)      ' node number
aryNOD(i, 2) = Mid$(buffer, 6, 1)      ' coordinate type
aryNOD(i, 3) = Mid$(buffer, 7, 1)      ' generation level
aryNOD(i, 4) = Mid$(buffer, 10, 1)     ' Secondary/Primary
aryNOD(i, 5) = Mid$(buffer, 11, 10)    ' x coordinate
aryNOD(i, 6) = Mid$(buffer, 21, 10)    ' y coordinate
aryNOD(i, 7) = Mid$(buffer, 31, 10)    ' z coordinate
aryNOD(i, 8) = Mid$(buffer, 41, 10)    ' TMASS(1)
aryNOD(i, 9) = Mid$(buffer, 51, 10)    ' TMASS(2)
aryNOD(i, 10) = Mid$(buffer, 61, 10)   ' TMASS(3)
aryNOD(i, 11) = Mid$(buffer, 71, 10)   ' TMASS(4)
```

End Sub

### Sub OutputData (buffer)

*'This routine extracts fields from the output data line*

```

Dim temp As Integer      'temporary variable

NPFREQ = CInt(Mid$(buffer, 1, 5))      ' frequency of output
frmHSMInput.txtNPFREQ = Mid$(buffer, 1, 5)
NPRU = CInt(Mid$(buffer, 6, 5))      ' number of motion output lines
frmHSMInput.txtNPRU = Mid$(buffer, 6, 5)
NPRS = CInt(Mid$(buffer, 11, 5))      ' number of stress output lines
frmHSMInput.txtNPRS = Mid$(buffer, 11, 5)
If Len(buffer) > 15 Then
    If Mid$(buffer, 16, 5) <> Space$(5) Then
        NPIC = CInt(Mid$(buffer, 16, 5))      ' number of complete output pictures
    End If
    frmHSMInput.txtNPIC = Mid$(buffer, 16, 5)
    If Mid$(buffer, 21, 5) <> Space$(5) Then
        NANG = CInt(Mid$(buffer, 21, 5))      ' used when ELMPLT is desired in output
    End If
    frmHSMInput.txtNANG = Mid$(buffer, 21, 5)
    frmHSMInput.txtNDREF = Mid$(buffer, 26, 5)
Else
    NPIC = 0
    NANG = 0
    frmHSMInput.txtNPIC = FormatNum(NPIC, 5)
    frmHSMInput.txtNANG = FormatNum(NANG, 5)
    frmHSMInput.txtNDREF = FormatNum(0, 5)
End If

End Sub

Sub ParameterLine (buffer)

    'This routine extracts fields from a parameter line

    frmHSMInput.txtNNODE = Mid$(buffer, 1, 5)
    NNODE = CInt(Mid$(buffer, 1, 5))      ' number of nodes in the model
    frmHSMInput.txtNPRI = Mid$(buffer, 6, 5)
    NPRI = CInt(Mid$(buffer, 6, 5))      ' number of primary nodes in the model
    frmHSMInput.txtNAXOR = Mid$(buffer, 11, 5)
    NAXOR = CInt(Mid$(buffer, 11, 5))      ' number of axis orientation modes in the model
    frmHSMInput.txtNELE = Mid$(buffer, 16, 5)
    NELE = CLng(Mid$(buffer, 16, 5))      ' number of elements in the model
    frmHSMInput.txtNUMMAT = Mid$(buffer, 21, 5)
    NUMMAT = CLng(Mid$(buffer, 21, 5))      ' number of different element section and material types
    frmHSMInput.txtNUMDIS = Mid$(buffer, 26, 5)
    NUMDIS = CInt(Mid$(buffer, 26, 5))      ' number of nodes at which any displacement components
    >>> are specified
    frmHSMInput.txtMXSTEP = Mid$(buffer, 31, 6)
    MXSTEP = CLng(Mid$(buffer, 31, 6))      ' number of time steps to be taken
    frmHSMInput.txtNDGREE = Mid$(buffer, 37, 4)
    NDGREE = CInt(Mid$(buffer, 37, 4))      ' number of degrees of freedom per node
    frmHSMInput.txtDELT = Mid$(buffer, 41, 10)
    'DELT = CDbl(Mid$(buffer, 41, 10))      ' time increment
    frmHSMInput.txtNODMAX = Mid$(buffer, 51, 5)
    NODMAX = Val(Mid$(buffer, 51, 5))      ' largest node number in model
    If NODMAX = 0 Then
        NODMAX = NNODE + NAXOR
        frmHSMInput.txtNODMAX = FormatNum(NODMAX, 5)
    End If

End Sub

Sub PictureLines (buffer, i)

    'This routine extracts fields from a output picture line

    aryPIC(i, 1) = Mid$(buffer, 1, 10)      ' time step at which complete output picture is desired
    aryPIC(i, 2) = Mid$(buffer, 11, 10)      ' KON (0-6)

End Sub

Sub PlaneID (filenumber, buffer, i)

    'This routine reads plane identification lines and

```

**'extracts the field data**

```

Dim j As Integer      'counter

Line Input #filenumber, buffer
aryNOP(i, 1) = Mid$(buffer, 1, 5)      ' primary node number designating the plane
aryNOP(i, 2) = Mid$(buffer, 6, 5)      ' total number of primary nodes of the model which may
»» contact plane I
aryNOP(i, 3) = Mid$(buffer, 11, 10)    ' direction cosine of acceleration vector with respect to
»» the global x-axis
aryNOP(i, 4) = Mid$(buffer, 21, 10)    ' direction cosine of acceleration vector with respect to
»» the global y-axis
aryNOP(i, 5) = Mid$(buffer, 31, 10)    ' direction cosine of acceleration vector with respect to
»» the global z-axis
aryNOP(i, 6) = Mid$(buffer, 41, 10)    ' linear stiffness of elastic plane I
aryNOP(i, 7) = Mid$(buffer, 51, 10)    ' cubic stiffness of elastic plane I
aryNOP(i, 8) = Mid$(buffer, 61, 10)    ' fraction of viscous damping for plane I

Line Input #filenumber, buffer
If Val(aryNOP(i, 2)) > 0 Then
    For j = 1 To Val(aryNOP(i, 2))
        aryNOP(i, j + 8) = Mid$(buffer, (j - 1) * 5 + 1, 5)
    Next j
Else
    aryNOP(i, 9) = Mid$(buffer, 1, 5)    'first primary node
    aryNOP(i, 10) = Mid$(buffer, 6, 5)   'last primary node
End If
Line Input #filenumber, buffer
aryNOP(i, 19) = Mid$(buffer, 1, 10)     'x coordinate for point 1
aryNOP(i, 20) = Mid$(buffer, 11, 10)    'y coordinate for point 1
aryNOP(i, 21) = Mid$(buffer, 21, 10)    'z coordinate for point 1
Line Input #filenumber, buffer
aryNOP(i, 22) = Mid$(buffer, 1, 10)     'x coordinate for point 2
aryNOP(i, 23) = Mid$(buffer, 11, 10)    'y coordinate for point 2
aryNOP(i, 24) = Mid$(buffer, 21, 10)    'z coordinate for point 2
Line Input #filenumber, buffer
aryNOP(i, 25) = Mid$(buffer, 1, 10)     'x coordinate for point 3
aryNOP(i, 26) = Mid$(buffer, 11, 10)    'y coordinate for point 3
aryNOP(i, 27) = Mid$(buffer, 21, 10)    'z coordinate for point 3
If KONTRL(14) = 1 Then
    Line Input #filenumber, buffer
End If

```

**End Sub**

**Sub PlotLines (filenumber, buffer)**

**'This routine reads deformation configuration plot lines  
'and extracts the field values**

```

Dim i As Integer      ' counter
Dim ICMND As Integer  ' number of command lines
Dim xmin As String    ' x' minimum value
Dim xmax As String    ' x' maximum value
Dim ymin As String    ' y' minimum value
Dim ymax As String    ' y' maximum value
Dim zmin As String    ' z' minimum value
Dim zmax As String    ' z' maximum value
Dim IDN As Integer    ' fields in command line
Dim NP As Integer     ' fields in command line
Dim ID As String      ' fields in command line
Dim str1 As String    ' temporary string variable
Dim CRLF As String    ' carriage return/line feed

```

CRLF = Chr\$(13) + Chr\$(10)

**'line 15A - plot choices**

```

Line Input #filenumber, buffer
PLTDCD = Val(Mid$(buffer, 1, 5))
PLTLAB = Val(Mid$(buffer, 6, 5))
IOPTPLT = Val(Mid$(buffer, 11, 5))
frmHSMInput.txtPLTDCD.Text = Mid$(buffer, 1, 5)
frmHSMInput.txtPLTLAB.Text = Mid$(buffer, 6, 5)
frmHSMInput.txtIOPTPLT.Text = Mid$(buffer, 11, 5)

```

**'line 15B - heading**

```
Line Input #filenumber, buffer
TITLE(1) = Mid$(buffer, 1, 10)
TITLE(2) = Mid$(buffer, 11, 10)
TITLE(3) = Mid$(buffer, 21, 10)
frmHSMInput.txtTitle2.Text = TITLE(1) + " " + TITLE(2) + " " + TITLE(3)
```

**'line 15C - coordinates to be plotted**

```
Line Input #filenumber, buffer
xmin = Mid$(buffer, 1, 10)
xmax = Mid$(buffer, 11, 10)
ymin = Mid$(buffer, 21, 10)
ymax = Mid$(buffer, 31, 10)
zmin = Mid$(buffer, 41, 10)
zmax = Mid$(buffer, 51, 10)
frmHSMInput.txtxmin = xmin
frmHSMInput.txtxmax = xmax
frmHSMInput.txtymin = ymin
frmHSMInput.txtymax = ymax
frmHSMInput.txtzmin = zmin
frmHSMInput.txtzmax = zmax
```

**'line 15D - size of plots and location of origin**

```
Line Input #filenumber, buffer
TOTALX = Val(Mid$(buffer, 1, 10))
TOTALY = Val(Mid$(buffer, 11, 10))
TOTALZ = Val(Mid$(buffer, 21, 10))
XNEG = Val(Mid$(buffer, 31, 10))
YNEG = Val(Mid$(buffer, 41, 10))
ZNEG = Val(Mid$(buffer, 51, 10))
ICMND = Val(Mid$(buffer, 61, 5))
frmHSMInput.txtTotalx.Text = Mid$(buffer, 1, 10)
frmHSMInput.txtTotaly.Text = Mid$(buffer, 11, 10)
frmHSMInput.txtTotalz.Text = Mid$(buffer, 21, 10)
frmHSMInput.txtxneg.Text = Mid$(buffer, 31, 10)
frmHSMInput.txtyneg.Text = Mid$(buffer, 41, 10)
frmHSMInput.txtzneg.Text = Mid$(buffer, 51, 10)
```

**'line 15E - Command Lines**

```
str1 = ""
For i = 1 To ICMND
    Line Input #filenumber, buffer
    IDN = Val(Mid$(buffer, 1, 2))
    NP = Val(Mid$(buffer, 3, 2))
    ID = Mid$(buffer, 5, 4)
    str1 = str1 + Mid$(buffer, 1, 2) + " " + Mid$(buffer, 3, 2) + " " + Mid$(buffer, 5, 4) + CRLF
Next i
frmHSMInput.txtCommand.Text = str1
```

**End Sub**

**Sub ProgramControlLine (buffer)**

**'This routine extracts fields from the program control line**

```
Dim temp As Integer      'temporary variable

KONTRL(1) = Val(Mid$(buffer, 1, 5))      ' global or local coordinate option for secondary
»» nodes
frmHSMInput.optKONTRL1(KONTRL(1)).Value = True
If Mid$(buffer, 1, 5) = Space$(5) Then
    frmHSMInput.optKONTRL1(0) = True
End If
KONTRL(2) = Val(Mid$(buffer, 6, 5))      ' print option
If KONTRL(2) = 0 Then
    frmHSMInput.chkKONTRL2.Value = True
End If
KONTRL(3) = Val(Mid$(buffer, 11, 5))      ' control parameter for initial body axes
frmHSMInput.optKONTRL3(KONTRL(3)).Value = True
KONTRL(4) = Val(Mid$(buffer, 16, 5))      ' number of sliding interface planes
frmHSMInput.txtKONTRL4 = Mid$(buffer, 16, 5)
KONTRL(5) = Val(Mid$(buffer, 21, 5))      ' beta parameter in the Newmark integration
frmHSMInput.txtKONTRL5 = Mid$(buffer, 21, 5)
KONTRL(6) = Val(Mid$(buffer, 26, 5))      ' number of secondary bodies for contact
frmHSMInput.txtKONTRL6 = Mid$(buffer, 26, 5)
KONTRL(7) = Val(Mid$(buffer, 31, 5))
```



```

' default node type
frmHSMInput.optKONTRL7(KONTRL(7)).Value = True
KONTRL(8) = Val(Mid$(buffer, 36, 5)) ' critical or viscous damping
frmHSMInput.optKONTRL8(KONTRL(8)).Value = True
KONTRL(9) = Val(Mid$(buffer, 41, 5)) ' type of integration
frmHSMInput.optKONTRL9(KONTRL(9)).Value = True
KONTRL(10) = Val(Mid$(buffer, 46, 5)) ' restart control
frmHSMInput.txtKONTRL10 = Mid$(buffer, 46, 5)
KONTRL(11) = Val(Mid$(buffer, 51, 5)) ' Q array output?
frmHSMInput.optKONTRL11(KONTRL(11)).Value = True
KONTRL(12) = Val(Mid$(buffer, 56, 5)) ' triangle place element
frmHSMInput.optKONTRL12(KONTRL(12)).Value = True
KONTRL(13) = Val(Mid$(buffer, 61, 5)) ' logical unit option
KONTRL(14) = Val(Mid$(buffer, 66, 5)) ' contact with seat back?
frmHSMInput.optKONTRL14(KONTRL(14)).Value = True
KONTRL(15) = Val(Mid$(buffer, 71, 5)) ' injury potential determined?
frmHSMInput.optKONTRL15(KONTRL(15)).Value = True
KONTRL(16) = Val(Mid$(buffer, 76, 5)) ' preejection pilot alignment simulation?
frmHSMInput.optKONTRL16(KONTRL(16)).Value = True

```

End Sub

Sub ReadInput (filenumber, buffer)

```

' This routine reads each line of the input file that involves input data:
' Title, parameters, control, material properties, nodes, displacement,
' cross-sectional geometries, x-bar and y-bar vectors, and 7F lines.
' Subroutines called: ParameterLine
' ProgramControlLine
' MaterialPropertyLine(1-3)
' NodalData
' Elements
' Displacement
' CrossSection
' Vectors

```

Dim i As Integer

' line 1 - TITLE

Line Input #filenumber, buffer

frmHSMInput.txtTITLE = buffer

' line 2 - Parameter Line

Line Input #filenumber, buffer

ParameterLine (buffer)

' line 3 - Program Control Line

Line Input #filenumber, buffer

ProgramControlLine (buffer)

' line 4 - Material Property Lines

frmHSMInput.tblMatProp.Rows = NUMMAT

ReDim aryMAT(NUMMAT, 14)

For i = 1 To NUMMAT

Line Input #filenumber, buffer

Call MaterialPropertyLine1(buffer, i)

Line Input #filenumber, buffer

Call MaterialPropertyLine2(buffer, i)

Line Input #filenumber, buffer

Call MaterialPropertyLine3(buffer, i)

Next i

frmHSMInput.tblMatProp.Refresh

' line 5 - Nodal Data Lines

frmHSMInput.tblNodalData.Rows = NNODE + NAXOR

ReDim aryNOD(NNODE + NAXOR, 11)

For i = 1 To NNODE + NAXOR

Line Input #filenumber, buffer

Call NodalData(buffer, i)

Next i

frmHSMInput.tblNodalData.Refresh

' line 6 - Element Data Lines

frmHSMInput.tblElements.Rows = NELE

ReDim aryELE(NELE, 16)

For i = 1 To NELE

Line Input #filenumber, buffer

```

    Call Elements(buffer, i)
Next i
frmHSMInput.tblElements.Refresh

' line 7 - Prescribed Displacement Lines
frmHSMInput.tblDisplace.Rows = NUMDIS
ReDim aryDIS(NUMDIS, 11)
For i = 1 To NUMDIS
    Line Input #filename, buffer
    Call Displacement(buffer, i)
Next i
frmHSMInput.tblDisplace.Refresh

'line 7B & 7C - Cross-Sectional Geometries Code
If KONTRL(9) <> 0 Then
    Line Input #filename, buffer
    NUMSEC = Val(Mid$(buffer, 1, 5))
    If NUMSEC <> 0 Then
        ReDim aryIPT(NUMSEC)
        If NUMSEC <= 16 Then
            Line Input #filename, buffer
            Call CrossSection(buffer)
        End If
    End If
End If

'line 7D & 7E - X-Bar, Y-Bar Vectors
If KONTRL(3) = 2 Then
    ReDim aryVEC(NPRI, 8)
    For i = 1 To NPRI
        Line Input #filename, buffer
        Call Vectors(buffer, i)
    Next i
End If

'line 7F
Line Input #filename, buffer
If Len(buffer) > 0 Then
    NLOAD = Val(Mid$(buffer, 1, 5))
    NIC = Val(Mid$(buffer, 6, 5))
Else
    NLOAD = 0
    NIC = 0
End If

```

**End Sub**

**Sub ReadInputFile ()**

```

' This routine shows an open file dialog and allows the user to select
' a .inp file.
' Subroutines called: ReadInput
'                      ReadOutput

Dim buffer As String * 80      ' buffer for file read
Dim filename As Integer       ' file handle

' erase arrays
Erase aryMAT
Erase aryNOD
Erase aryELE
Erase aryDIS
Erase aryIPT
Erase aryVEC
Erase aryMOT
Erase arySTR
Erase aryPIC
Erase aryFSL
Erase aryNOP
Erase arySPN

' display open file dialog
frmHSMInput.dlgFile.Filter = " (*.inp) | *.inp "
frmHSMInput.dlgFile.Action = 1
If frmHSMInput.dlgFile.FileName <> "" Then

```

```

        Infile = frmHSMInput.dlgFile.FileName
        'open file selected
        filenumber = FreeFile
        Open frmHSMInput.dlgFile.FileName For Input Access Read As #filenumber Len = 80
        On Error GoTo ErrorHandler
        'read input data
        Call ReadInput(filenumber, buffer)
        'read output data
        Call ReadOutput(filenumber, buffer)
        Close filenumber
    End If
    Exit Sub

ErrorHandler:
    If Err = 62 Then Exit Sub
    Resume Next

```

**End Sub**

**Sub ReadOutput (filenumber, buffer)**

```

    'This routine reads each line of the input file that involves output
    'specification:
    ' Output Control Lines, Motion Output Lines, Stress Output Lines,
    ' Complete Output Picture Lines, Deformation Amplification Factors,
    ' ICIF Data Lines, Plane Identification Lines, Spinal Injury Function
    ' Lines, and Deformation Configuration Plot Lines.
    ' Subroutines called: OutputData
    '                        MotionLines
    '                        StressLines
    '                        PictureLines
    '                        PlotInfo
    '                        ICIF
    '                        PlaneID
    '                        SpinalInjury
    '                        PlotLines

    Dim i As Integer        'counter
    Dim j As Integer        'counter

    On Error GoTo ErrorHandler

    'line 8 - Output Control Line
    Line Input #filenumber, buffer
    Call OutputData(buffer)

    'line 9 - Motion Output Lines
    If NPRU > 0 Then
        frmHSMInput.tblMotion.Rows = NPRU
        ReDim aryMOT(NPRU, 5)
        For i = 1 To NPRU
            Line Input #filenumber, buffer
            Call MotionLines(buffer, i)
        Next i
        frmHSMInput.tblMotion.Refresh
    End If

    'line 10 - Stress Output Lines
    If NPRS > 0 Then
        frmHSMInput.tblStress.Rows = NPRS
        ReDim arySTR(NPRS, 7)
        For i = 1 To NPRS
            Line Input #filenumber, buffer
            Call StressLines(buffer, i)
        Next i
        frmHSMInput.tblStress.Refresh
    Else
        frmHSMInput.tblStress.Rows = NPRS
        frmHSMInput.tblStress.Refresh
    End If

    'line 11 - Complete Output Picture Lines
    If NPIC > 0 Then
        frmHSMInput.tblPicLines.Rows = NPIC
        ReDim aryPIC(NPIC, 2)
    End If

```

```

    For i = 1 To NPIC
        Line Input #filenumber, buffer
        Call PictureLines(buffer, i)
    Next i
    frmHSMInput.tblPicLines.Refresh
Else
    frmHSMInput.tblPicLines.Rows = NPIC
    frmHSMInput.tblPicLines.Refresh
End If

'line 8A & 8B - Deformation Amplification Factor
If NANG > 0 Then
    For i = 1 To NANG + 1
        Line Input #filenumber, buffer
        'Call PlotInfo(buffer, i)
    Next i
End If

'line 12 - ICIF Data Lines
Line Input #filenumber, buffer
NPTS = Val(Mid$(buffer, 1, 5))
If Mid$(buffer, 11, 10) <> Space(0) Then
    F1 = CDBl(Mid$(buffer, 11, 10))
End If
If Mid$(buffer, 21, 10) <> Space(0) Then
    F2 = CDBl(Mid$(buffer, 21, 10))
End If
frmHSMInput.txtF1 = Mid$(buffer, 11, 10)
frmHSMInput.txtF2 = Mid$(buffer, 21, 10)
frmHSMInput.tblFuncSpec.Rows = NPTS
ReDim aryFSL(NPTS, 3)
For i = 1 To NPTS
    Line Input #filenumber, buffer
    Call ICIF(buffer, i)
Next i
frmHSMInput.tblFuncSpec.Refresh

'line 13 - Plane Identification Line
ReDim aryNOP(KONTRL(4), 27)
If KONTRL(4) <> 0 Then
    frmHSMInput.tblPlaneLines.Rows = KONTRL(4)
    For i = 1 To KONTRL(4)
        Call PlaneID(filenumber, buffer, i)
    Next i
    frmHSMInput.tblPlaneLines.Refresh
Else
    frmHSMInput.tblPlaneLines.Rows = KONTRL(4)
    frmHSMInput.tblPlaneLines.Refresh
End If

'line 14 - Spinal Injury Function Lines
If Not EOF(filenumber) Then
    Call SpinalInjury(filenumber, buffer)
Else
    frmHSMInput.txtFACT.Text = ""
    frmHSMInput.txtNSTART.Text = ""
    frmHSMInput.txtISYM.Text = ""
    frmHSMInput.tblSpine.Rows = 0
    frmHSMInput.tblSpine.Refresh
End If

'line 15 - Deformation Configuration Plot Lines
If Not EOF(filenumber) Then
    Call PlotLines(filenumber, buffer)
Else
    frmHSMInput.txtPLTDCD.Text = ""
    frmHSMInput.txtPLTLAB.Text = ""
    frmHSMInput.txtIOPTPLT.Text = ""
    frmHSMInput.txtTitle2.Text = ""
    frmHSMInput.txtxmin.Text = ""
    frmHSMInput.txtxmax.Text = ""
    frmHSMInput.txtymin.Text = ""
    frmHSMInput.txtymax.Text = ""
    frmHSMInput.txtzmin.Text = ""
    frmHSMInput.txtzmax.Text = ""
    frmHSMInput.txtTotalx.Text = ""
    frmHSMInput.txtTotaly.Text = ""

```

```

        frmHSMInput.txtTotalz.Text = ""
        frmHSMInput.txtxneg.Text = ""
        frmHSMInput.txtyneg.Text = ""
        frmHSMInput.txtzneg.Text = ""
        frmHSMInput.txtCommand.Text = ""
    End If

    Close #filenumber
    Exit Sub

ErrorHandler0:
    If Err = 62 Then Exit Sub
    Resume Next

End Sub

Sub SpinalInjury (filenumber, buffer)

    'This routine reads spinal injury lines and extracts the field values

    Dim i As Integer      'counter
    Dim j As Integer      'counter
    Dim k As Integer      'counter

    'line 14A
    Line Input #filenumber, buffer
    JSTART = Val(Mid$(buffer, 1, 10))    ' bottommost vertebral level
    JEND = Val(Mid$(buffer, 11, 10))     ' uppermost vertebral level
    NB = JEND - JSTART
    frmHSMInput.tblSpine.Rows = NB
    ReDim arySPN(NB, 4)
    For i = 1 To NB
        arySPN(i, 1) = Str$((JSTART - 1) + i)
    Next i

    'line 14B - Axial Compression
    j = 0
    k = 0
    Do While k < NB
        Line Input #filenumber, buffer
        For i = 1 To 16
            arySPN(i + j, 2) = Mid$(buffer, (i * 5) - 4, 5)
            k = i + j
            If k = NB Then Exit For
        Next i
        j = k
    Loop

    'line 14C - Lateral Bending Moment
    j = 0
    k = 0
    Do While k < NB
        Line Input #filenumber, buffer
        For i = 1 To 16
            arySPN(i + j, 3) = Mid$(buffer, (i * 5) - 4, 5)
            k = i + j
            If k = NB Then Exit For
        Next i
        j = k
    Loop

    'line 14D - Anterior-posterior Bending Moment
    j = 0
    k = 0
    Do While k < NB
        Line Input #filenumber, buffer
        For i = 1 To 16
            arySPN(i + j, 4) = Mid$(buffer, (i * 5) - 4, 5)
            k = i + j
            If k = NB Then Exit For
        Next i
        j = k
    Loop

    'line 14E - Exponential factor and symmetry

```

```

Line Input #filenumber, buffer
frmHSMInput.txtFACT.Text = Mid$(buffer, 1, 10)
frmHSMInput.txtNSTART.Text = Mid$(buffer, 11, 5)
frmHSMInput.txtISYM.Text = Mid$(buffer, 16, 5)

```

**End Sub**

**Sub StressLines (buffer, i)**

*'This routine extracts fields from a stress output line*

```

arySTR(i, 1) = Mid$(buffer, 1, 5)      ' element number
arySTR(i, 2) = Mid$(buffer, 6, 1)      ' I1
arySTR(i, 3) = Mid$(buffer, 7, 1)      ' I2
arySTR(i, 4) = Mid$(buffer, 8, 1)      ' I3
arySTR(i, 5) = Mid$(buffer, 9, 1)      ' M
arySTR(i, 6) = Mid$(buffer, 10, 1)     ' N
arySTR(i, 7) = Mid$(buffer, 21, 40)    ' label

```

**End Sub**

**Sub Vectors (buffer, i)**

*'This routine extracts fields from the X-Bar and Y-Bar Vector lines*

```

aryVEC(i, 1) = Mid$(buffer, 1, 5)      ' node number
aryVEC(i, 2) = Mid$(buffer, 5, 5)      ' space
aryVEC(i, 3) = Mid$(buffer, 11, 10)     ' EULCO(1,1)
aryVEC(i, 4) = Mid$(buffer, 21, 10)     ' EULCO(2,1)
aryVEC(i, 5) = Mid$(buffer, 31, 10)     ' EULCO(3,1)
aryVEC(i, 6) = Mid$(buffer, 41, 10)     ' EULCO(1,2)
aryVEC(i, 7) = Mid$(buffer, 51, 10)     ' EULCO(2,2)
aryVEC(i, 8) = Mid$(buffer, 61, 10)     ' EULCO(3,2)

```

**End Sub**

Option Explicit

**Function ControlOut (buffer) As String**

*'This function sets the control line buffer based  
'on the values on the input screen.*

```

Dim i As Integer          'counter
Dim temp As Integer       'temporary variable

' global or local coordinate option for secondary nodes
If frmHSMInput.optKONTRL1(0).Value = True Then
    Mid$(buffer, 1, 5) = "    0"
Else
    Mid$(buffer, 1, 5) = "    1"
End If

' print option
If frmHSMInput.chkKONTRL2.Value = True Then
    Mid$(buffer, 6, 5) = "    0"
Else
    Mid$(buffer, 6, 5) = "    1"
End If

' control parameter for initial body axes
For i = 0 To 2
    If frmHSMInput.optKONTRL3(i).Value = True Then
        Mid$(buffer, 11, 5) = "    " & Str$(i)
    End If
Next i

' number of sliding interface planes
Mid$(buffer, 16, 5) = FormatNum(Val(frmHSMInput.txtKONTRL4.Text), 5)
' beta parameter in the Newmark integration
Mid$(buffer, 21, 5) = FormatNum(Val(frmHSMInput.txtKONTRL5.Text), 5)
' number of secondary bodies for contact
Mid$(buffer, 26, 5) = FormatNum(Val(frmHSMInput.txtKONTRL6.Text), 5)

' default node type
If frmHSMInput.optKONTRL7(0).Value = True Then
    Mid$(buffer, 31, 5) = "    0"
Else
    Mid$(buffer, 31, 5) = "    1"
End If

' critical or viscous damping
If frmHSMInput.optKONTRL8(0).Value = True Then
    Mid$(buffer, 36, 5) = "    0"
Else
    Mid$(buffer, 36, 5) = "    1"
End If

' type of integration
If frmHSMInput.optKONTRL9(0).Value = True Then
    Mid$(buffer, 41, 5) = "    0"
Else
    Mid$(buffer, 41, 5) = "    1"
End If

' restart control
Mid$(buffer, 46, 5) = Format$(frmHSMInput.txtKONTRL10.Text, "#####")

' Q array output?
For i = 0 To 2
    If frmHSMInput.optKONTRL11(i).Value = True Then
        Mid$(buffer, 51, 5) = "    " & Str$(i)
    End If
Next i

' triangle place element
If frmHSMInput.optKONTRL12(0).Value = True Then

```

```

    Mid$(buffer, 56, 5) = "    0"
Else
    Mid$(buffer, 56, 5) = "    1"
End If

' logical unit option
Mid$(buffer, 61, 5) = "    1"

' contact with seat back?
If frmHSMInput.optKONTRL14(0).Value = True Then
    Mid$(buffer, 66, 5) = "    0"
Else
    Mid$(buffer, 66, 5) = "    1"
End If

' injury potential determined?
If frmHSMInput.optKONTRL15(0).Value = True Then
    Mid$(buffer, 71, 5) = "    0"
Else
    Mid$(buffer, 71, 5) = "    1"
End If

' preejection pilot alignment simulation?
If frmHSMInput.optKONTRL16(0).Value = True Then
    Mid$(buffer, 76, 5) = "    0"
Else
    Mid$(buffer, 76, 5) = "    1"
End If
ControlOut = buffer

```

**End Function**

**Sub CrossOut (filename, buffer)**

```

' This routine builds the buffer and writes cross sectional geometries
' lines to the input file.

```

```

Dim i As Integer      'counter

Mid$(buffer, 1, 5) = FormatNum(NUMSEC, 5)
Print #filename, buffer
buffer = Space(80)
If NUMSEC <> 0 Then
    If NUMSEC < 16 Then
        For i = 1 To NUMSEC
            Mid$(buffer, (i - 1) * 5 + 1, 5) = aryIPT(i)
        Next i
        Print #filename, buffer
        buffer = Space(80)
    End If
End If

```

**End Sub**

**Sub DisplaceOut (filename, buffer)**

```

' This routine builds the buffer and writes displacement
' lines to the input file.

```

```

Dim i As Integer      'counter

For i = 1 To NUMDIS
    Mid$(buffer, 1, 4) = aryDIS(i, 1)  ' node number
    Mid$(buffer, 5, 1) = aryDIS(i, 2)  ' translational x
    Mid$(buffer, 6, 1) = aryDIS(i, 3)  ' translational y
    Mid$(buffer, 7, 1) = aryDIS(i, 4)  ' translational z
    Mid$(buffer, 8, 1) = aryDIS(i, 5)  ' rotation x
    Mid$(buffer, 9, 1) = aryDIS(i, 6)  ' rotation y
    Mid$(buffer, 10, 1) = aryDIS(i, 7) ' rotation z
    Print #filename, buffer
    buffer = Space(80)
Next i

```

**End Sub**



### Sub ElementOut (filenumber, buffer)

*'This routine builds the buffer and writes element  
'lines to the input file.*

```
Dim i As Integer      'counter

For i = 1 To NELE
    Mid$(buffer, 1, 5) = aryELE(i, 1)      ' node number
    Mid$(buffer, 6, 5) = aryELE(i, 2)      ' Node I
    Mid$(buffer, 11, 5) = aryELE(i, 3)      ' Node J
    Mid$(buffer, 16, 5) = aryELE(i, 4)      ' primary node for I
    Mid$(buffer, 21, 5) = aryELE(i, 5)      ' primary node for J
    Mid$(buffer, 26, 5) = aryELE(i, 6)      ' not used
    Mid$(buffer, 31, 5) = aryELE(i, 7)      ' not used
    Mid$(buffer, 36, 5) = aryELE(i, 8)      ' not used
    Mid$(buffer, 41, 5) = aryELE(i, 9)      ' Node K
    Mid$(buffer, 46, 5) = aryELE(i, 10)     ' material type
    Mid$(buffer, 51, 5) = aryELE(i, 11)     ' element type
    Mid$(buffer, 56, 5) = aryELE(i, 12)     ' not used
    Mid$(buffer, 61, 5) = aryELE(i, 13)     ' number of sliding node pairs
    Mid$(buffer, 66, 5) = aryELE(i, 14)     ' not used
    Mid$(buffer, 71, 5) = aryELE(i, 15)     ' not used
    Mid$(buffer, 76, 5) = aryELE(i, 16)     ' level
    Print #filenumber, buffer
    buffer = Space(80)
Next i
```

End Sub

### Sub ICIFOut (filenumber, buffer)

*'This routine builds the buffer and writes ICIF  
'lines to the input file.*

```
Dim i As Integer      'counter

Mid$(buffer, 1, 5) = FormatNum(NPTS, 5)
Mid$(buffer, 6, 5) = Space(5)
Mid$(buffer, 11, 10) = frmHSMInput.txtF1.Text
Mid$(buffer, 21, 10) = frmHSMInput.txtF2.Text
Print #filenumber, buffer
buffer = Space(80)
For i = 1 To NPTS
    Mid$(buffer, 1, 10) = aryFSL(i, 1)      ' Time t
    Mid$(buffer, 11, 10) = aryFSL(i, 2)      ' motion function value at time t
    Mid$(buffer, 21, 10) = aryFSL(i, 3)      ' derivative of function at time t
    Print #filenumber, buffer
    buffer = Space(80)
Next i
```

End Sub

### Sub MaterialOut (filenumber, buffer)

*'This routine builds the buffer and writes material properties  
'lines to the input file.*

```
Dim i As Integer      'counter

' line 4 - Material Property Lines
For i = 1 To NUMMAT
    Mid$(buffer, 1, 5) = aryMAT(i, 1)      ' MTYP
    Mid$(buffer, 6, 5) = aryMAT(i, 2)      ' LTYP
    Print #filenumber, buffer
    buffer = Space(80)
    Mid$(buffer, 1, 10) = aryMAT(i, 3)      ' E(1,MTYP)
    Mid$(buffer, 11, 10) = aryMAT(i, 4)      ' E(2,MTYP)
    Mid$(buffer, 21, 10) = aryMAT(i, 5)      ' E(3,MTYP)
    Mid$(buffer, 31, 10) = aryMAT(i, 6)      ' E(4,MTYP)
    Mid$(buffer, 41, 10) = aryMAT(i, 7)      ' E(5,MTYP)
    Mid$(buffer, 51, 10) = aryMAT(i, 8)
```

```

' E(6,MTYP)
Print #filename, buffer
buffer = Space(80)
Mid$(buffer, 1, 10) = aryMAT(i, 9)      ' E(7,MTYP)
Mid$(buffer, 11, 10) = aryMAT(i, 10)     ' E(8,MTYP)
Mid$(buffer, 21, 10) = aryMAT(i, 11)     ' E(9,MTYP)
Mid$(buffer, 31, 10) = aryMAT(i, 12)     ' E(10,MTYP)
Mid$(buffer, 41, 10) = aryMAT(i, 13)     ' E(11,MTYP)
Mid$(buffer, 51, 10) = aryMAT(i, 14)     ' E(12,MTYP)
Print #filename, buffer
buffer = Space(80)
Next i

```

End Sub

Sub MotionOut (filename, buffer)

*'This routine builds the buffer and writes motion  
'lines to the input file.*

```

Dim i As Integer

If NPRU > 0 Then
For i = 1 To NPRU
Mid$(buffer, 1, 7) = aryMOT(i, 1)      ' node number
Mid$(buffer, 8, 1) = aryMOT(i, 2)      ' component number of kinematic variable to be output
Mid$(buffer, 9, 1) = aryMOT(i, 3)      ' displacement, velocity, or acceleration (0,1,2)
Mid$(buffer, 10, 1) = aryMOT(i, 4)     ' plot control (0-4)
Mid$(buffer, 21, 40) = aryMOT(i, 5)    ' label
Print #filename, buffer
buffer = Space(80)
Next i
End If

```

End Sub

Sub NodeOut (filename, buffer)

*'This routine builds the buffer and writes node  
'lines to the input file.*

```

Dim i As Integer      'counter

For i = 1 To NNODE + NAXOR
Mid$(buffer, 1, 5) = aryNOD(i, 1)      ' node number
Mid$(buffer, 6, 1) = aryNOD(i, 2)      ' coordinate type
Mid$(buffer, 7, 1) = aryNOD(i, 3)      ' generation level
Mid$(buffer, 10, 1) = aryNOD(i, 4)     ' Secondary/Primary
Mid$(buffer, 11, 10) = aryNOD(i, 5)    ' x coordinate
Mid$(buffer, 21, 10) = aryNOD(i, 6)    ' y coordinate
Mid$(buffer, 31, 10) = aryNOD(i, 7)    ' z coordinate
Mid$(buffer, 41, 10) = aryNOD(i, 8)    ' TMASS(1)
Mid$(buffer, 51, 10) = aryNOD(i, 9)    ' TMASS(2)
Mid$(buffer, 61, 10) = aryNOD(i, 10)   ' TMASS(3)
Mid$(buffer, 71, 10) = aryNOD(i, 11)   ' TMASS(4)
Print #filename, buffer
buffer = Space(80)
Next i

```

End Sub

Function OutputOut (buffer) As String

*'This routine builds the buffer and writes the output  
'line to the input file.*

```

' frequency of output
Mid$(buffer, 1, 5) = frmHSMInput.txtNPFREQ
' number of motion output lines
Mid$(buffer, 6, 5) = frmHSMInput.txtNPRU
' number of stress output lines
Mid$(buffer, 11, 5) = frmHSMInput.txtNPRS

```

```

' number of complete output pictures
Mid$(buffer, 16, 5) = frmHSMInput.txtNPIC
' used when ELMPLT is desired in output
Mid$(buffer, 21, 5) = frmHSMInput.txtNANG
' primary node for printer plots
Mid$(buffer, 26, 5) = frmHSMInput.txtNDREF
OutputOut = buffer

```

End Function

Function ParameterOut (buffer) As String

```

' This routine builds the buffer and writes the parameter
' line to the input file.

' number of nodes in the model
Mid$(buffer, 1, 5) = frmHSMInput.txtNNODE
' number of primary nodes in the model
Mid$(buffer, 6, 5) = frmHSMInput.txtNPRI
' number of axis orientation modes in the model
Mid$(buffer, 11, 5) = frmHSMInput.txtNAXOR
' number of elements in the model
Mid$(buffer, 16, 5) = frmHSMInput.txtNELE
' number of different element section and material types
Mid$(buffer, 21, 5) = frmHSMInput.txtNUMMAT
' number of nodes at which any displacement components are specified
Mid$(buffer, 26, 5) = frmHSMInput.txtNUMDIS
' number of time steps to be taken
Mid$(buffer, 31, 6) = frmHSMInput.txtMXSTEP
' number of degrees of freedom per node
Mid$(buffer, 37, 4) = frmHSMInput.txtNDGREE
' time increment
Mid$(buffer, 41, 10) = frmHSMInput.txtDELT
' largest node number in model
Mid$(buffer, 51, 5) = frmHSMInput.txtNODMAX
ParameterOut = buffer

```

End Function

Sub PictureOut (filenumber, buffer)

```

' This routine builds the buffer and writes picture
' lines to the input file.

Dim i As Integer      'counter

If NPIC > 0 Then
    For i = 1 To NPIC
        Mid$(buffer, 1, 10) = aryPIC(i, 1) ' time step at which complete output picture is desired
        Mid$(buffer, 11, 10) = aryPIC(i, 2) ' KON (0-6)
        Print #filenumber, buffer
        buffer = Space(80)
    Next i
End If

```

End Sub

Sub PlaneOut (filenumber, buffer)

```

' This routine builds the buffer and writes plane identification
' lines to the input file.

Dim i As Integer      'counter
Dim j As Integer      'counter

For i = 1 To KONTRL(4)
    ' line 13 - Plane Identification Line
    Mid$(buffer, 1, 5) = aryNOP(i, 1) ' primary node number designating the plane
    Mid$(buffer, 6, 5) = aryNOP(i, 2) ' total number of primary nodes of the model which may
    »» contact plane I
    Mid$(buffer, 11, 10) = aryNOP(i, 3) ' direction cosine of acceleration vector with respect
    »» to the global x-axis
    Mid$(buffer, 21, 10) = aryNOP(i, 4)

```

```

' direction cosine of acceleration vector with respect
»» to the global y-axis
Mid$(buffer, 31, 10) = aryNOP(i, 5)
' direction cosine of acceleration vector with respect
»» to the global z-axis
Mid$(buffer, 41, 10) = aryNOP(i, 6)
' linear stiffness of elastic plane I
Mid$(buffer, 51, 10) = aryNOP(i, 7)
' cubic stiffness of elastic plane I
Mid$(buffer, 61, 10) = aryNOP(i, 8)
' fraction of viscous damping for plane I
Print #filenumber, buffer
buffer = Space(80)
' line 13A - Contacting Primary Node Numbers Line
If Val(aryNOP(i, 2)) > 0 Then
  For j = 1 To Val(aryNOP(i, 2))
    Mid$(buffer, (j - 1) * 5 + 1, 5) = aryNOP(i, 8 + j)
  Next j
Else
  Mid$(buffer, 1, 5) = aryNOP(i, 9)
  Mid$(buffer, 6, 5) = aryNOP(i, 10)
End If
Print #filenumber, buffer
buffer = Space(80)
Mid$(buffer, 1, 10) = aryNOP(i, 19)
Mid$(buffer, 11, 10) = aryNOP(i, 20)
Mid$(buffer, 21, 10) = aryNOP(i, 21)
Print #filenumber, buffer
buffer = Space(80)
Mid$(buffer, 1, 10) = aryNOP(i, 22)
Mid$(buffer, 11, 10) = aryNOP(i, 23)
Mid$(buffer, 21, 10) = aryNOP(i, 24)
Print #filenumber, buffer
buffer = Space(80)
Mid$(buffer, 1, 10) = aryNOP(i, 25)
Mid$(buffer, 11, 10) = aryNOP(i, 26)
Mid$(buffer, 21, 10) = aryNOP(i, 27)
Print #filenumber, buffer
buffer = Space(80)
If KONTRL(14) = 1 Then
  'Line Input #filenumber, buffer
End If
Next i

```

End Sub

Sub PlotOut (filenumber, buffer)

```

' This routine builds the buffer and writes plot
' lines to the input file.

```

```

Dim i As Integer      'counter
If NANG > 0 Then
  For i = 1 To NANG + 1
    ' build buffer
    Print #filenumber, buffer
    buffer = Space(80)
  Next i
End If

```

End Sub

Sub SaveInputFile ()

```

' This routine displays a file selection dialog and saves the input file
' Subroutines called: WriteInput
'                      WriteOutput

```

```

Dim buffer As String * 80      ' record buffer
Dim filenumber As Integer      ' file handle

frmHSMInput.dlgFile.Filter = " (*.inp) | *.inp "
frmHSMInput.dlgFile.Action = 2
If frmHSMInput.dlgFile.Filename <> "" Then
  filenumber = FreeFile
  Open frmHSMInput.dlgFile.Filename For Output Access Write As #filenumber Len = 80
  Call WriteInput(filenumber, buffer)

```

```

    Call WriteOutput(filename, buffer)
End If
Close #filename
Exit Sub

```

**End Sub**

**Sub StressOut (filename, buffer)**

```

' This routine builds the buffer and writes stress
' lines to the input file.

```

```

Dim i As Integer      'counter

If NPRS > 0 Then
    For i = 1 To NPRS
        Mid$(buffer, 1, 5) = arySTR(i, 1)    ' element number
        Mid$(buffer, 6, 1) = arySTR(i, 2)    ' I1
        Mid$(buffer, 7, 1) = arySTR(i, 3)    ' I2
        Mid$(buffer, 8, 1) = arySTR(i, 4)    ' I3
        Mid$(buffer, 9, 1) = arySTR(i, 5)    ' M
        Mid$(buffer, 10, 1) = arySTR(i, 6)    ' N
        Mid$(buffer, 21, 40) = arySTR(i, 7)   ' label
        Print #filename, buffer
        buffer = Space(80)
    Next i
End If

```

**End Sub**

**Sub VectorOut (filename, buffer)**

```

' This routine builds the buffer and writes vector
' lines to the input file.

```

```

Dim i As Integer      'counter

For i = 1 To NPRI
    Mid$(buffer, 1, 5) = aryVEC(i, 1)        ' node number
    Mid$(buffer, 5, 5) = aryVEC(i, 2)        ' space
    Mid$(buffer, 11, 10) = aryVEC(i, 3)      ' EULCO(1,1)
    Mid$(buffer, 21, 10) = aryVEC(i, 4)      ' EULCO(2,1)
    Mid$(buffer, 31, 10) = aryVEC(i, 5)      ' EULCO(3,1)
    Mid$(buffer, 41, 10) = aryVEC(i, 6)      ' EULCO(1,2)
    Mid$(buffer, 51, 10) = aryVEC(i, 7)      ' EULCO(2,2)
    Mid$(buffer, 61, 10) = aryVEC(i, 8)      ' EULCO(3,2)
    Print #filename, buffer
    buffer = Space(80)
Next i

```

**End Sub**

**Sub WriteInput (filename, buffer)**

```

' This routine calls routines which write the input data to the
' input file.

```

```

' Subroutines called:  ParameterOut
'                      ControlOut
'                      MaterialOut
'                      NodeOut
'                      ElementOut
'                      DisplaceOut
'                      CrossOut
'                      VectorOut

```

```

Dim i As Integer      'counter

' line 1 - TITLE
buffer = frmHSMInput.txtTITLE
Print #filename, buffer
buffer = Space(80)
' line 2 - Parameter Line

```

```

buffer = ParameterOut(buffer)
Print #filenumber, buffer
buffer = Space(80)
' line 3 - Program Control Line
buffer = ControlOut(buffer)
Print #filenumber, buffer
buffer = Space(80)

' line 4 - Material Property Lines
Call MaterialOut(filenumber, buffer)
' line 5 - Nodal Data Lines
Call NodeOut(filenumber, buffer)
' line 6 - Element Data Lines
Call ElementOut(filenumber, buffer)
' line 7 - Prescribed Displacement Lines
Call DisplaceOut(filenumber, buffer)

'line 7B & 7C - Cross-Sectional Geometries Code
If KONTRL(9) <> 0 Then
    Call CrossOut(filenumber, buffer)
End If

'line 7D & 7E - X-Bar, Y-Bar Vectors
If KONTRL(3) = 2 Then
    Call VectorOut(filenumber, buffer)
End If

'line 7F
If NLOAD <> 0 Then
    Mid$(buffer, 1, 5) = FormatNum(NLOAD, 5)
    Mid$(buffer, 6, 5) = FormatNum(NIC, 5)
    Print #filenumber, buffer
Else
    Print #filenumber,
End If

```

End Sub

Sub WriteOutput (filenumber, buffer)

```

' This routine calls routines which write the output specifications to the
' input file.
' Subroutines called: MotionOut
'                      StressOut
'                      PictureOut
'                      PlotOut
'                      ICIFOut
'                      PlaneOut
'                      CrossOut
'                      VectorOut

'line 8 - Output Control Line
buffer = OutputOut(buffer)
Print #filenumber, buffer
buffer = Space(80)

'line 9 - Motion Output Lines
Call MotionOut(filenumber, buffer)

'line 10 - Stress Output Lines
Call StressOut(filenumber, buffer)

'line 11 - Complete Output Picture Lines
Call PictureOut(filenumber, buffer)

'line 8A & 8B - Deformation Amplification Factor
Call PlotOut(filenumber, buffer)

'line 12 - ICIF Data Lines
Call ICIFOut(filenumber, buffer)

'line 13 - Plane Identification Line
If KONTRL(4) <> 0 Then
    Call PlaneOut(filenumber, buffer)
End If

```



Option Explicit

Global Plotval() As Double     '*array of plot data*  
Global ttitles() As String     '*array of plot titles*

### Function ConvD (str1)

*'function to convert a string to a double precision number*  
*'(handles exponential notation)*

Dim n As Integer             '*position in string*  
Dim expon As Integer        '*exponent portion of string*  
Dim frac As Double          '*fractional portion of string*  
Dim num As Double            '*value to return*

n = InStr(1, str1, "D")  
If n <> 0 Then  
    expon = Val(Mid\$(str1, n + 1, 3))  
    frac = Val(Mid\$(str1, 1, n - 1))  
    num = frac \* 10 ^ expon  
Else  
    num = Val(str1)  
End If  
ConvD = num

End Function

### Sub PlotData (NPRU, numpts)

*'create data plots using Chart FX*  
*' NPRU = number of plots*  
*' numpts = number of data points*

Dim i As Integer            '*counter*  
Dim j As Integer            '*counter*  
Dim tempmin As Double        '*temporary variable for maximum data point*  
Dim tempmax As Double        '*temporary variable for minimum data point*

For j = 1 To NPRU  
    tempmin = Plotval(1, j + 1)  
    tempmax = Plotval(1, j + 1)  
    '*open communication channel to pass data*  
    frmPlot.Chart1.OpenData(COD\_VALUES) = CHART\_ML(1, 100)  
    '*setup axes and titles*  
    frmPlot.Chart1.ThisSerie = 0  
    frmPlot.Chart1.SerLeg(0) = "Time"  
    frmPlot.Chart1.LegStyle = CL\_HIDELEG  
    frmPlot.Chart1.FixedGap = 2  
    frmPlot.Chart1.Title(CHART\_TOPTIT) = ttitles(j)  
    For i = 1 To numpts  
        '*pass data values*  
        frmPlot.Chart1.Value(i - 1) = Plotval(i, j + 1)  
        If Plotval(i, j + 1) > tempmax Then tempmax = Plotval(i, j + 1)  
        If Plotval(i, j + 1) < tempmin Then tempmin = Plotval(i, j + 1)  
    Next i  
    '*setup axes minimums and maximums*  
    frmPlot.Chart1.Adm(CSA\_MIN) = tempmin  
    frmPlot.Chart1.Adm(CSA\_MAX) = tempmax  
    frmPlot.Chart1.Adm(CSA\_XMAX) = numpts  
    frmPlot.Chart1.CloseData(COD\_VALUES) = 0  
    '*set command button caption*  
    If j < NPRU Then  
        frmPlot.cmdClose.Caption = "Next"  
    Else  
        frmPlot.cmdClose.Caption = "Close"  
    End If  
    frmPlot.Show 1  
Next j

End Sub



## Sub ReadData (filenumber, buffer)

```

' read output file and store array values for plotting
'   filenumber = file handle
'   buffer      = buffer for file record
'   subroutines called: PlotData

Dim accum As Double      'time counter
Dim i As Integer         'counter
Dim j As Integer         'counter
Dim k As Integer         'counter
Dim stepcount As Long    'step counter
Dim numpts As Integer    'number of data points
Dim MXSTEP As Integer    'maximum number of steps
Dim NPFREQ As Integer    'output frequency
Dim ICMND As Integer     'command line

' read title
Line Input #filenumber, buffer

' read length of Q array
Line Input #filenumber, buffer

' read parameters
Line Input #filenumber, buffer      'number of nodes
  NNODE = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'number of primary nodes
  NPRI = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'number of axis orientation nodes
  NAXOR = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'number of elements
  NELE = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'number of materials
  NUMMAT = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'number of fixed nodes
  NUMDIS = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'number of increments
  MXSTEP = Val(Mid$(buffer, 43, 10))
Line Input #filenumber, buffer      'degrees of freedom
Line Input #filenumber, buffer      'time increment
  DELT = ConvD(Mid$(buffer, 43, 16))
Line Input #filenumber, buffer      'maximum node number
Line Input #filenumber, buffer      '2 blank lines
Line Input #filenumber, buffer

' read controls
Line Input #filenumber, buffer      'KONTRL(1)
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(2)
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(3)
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(4)
  KONTRL(4) = Val(Mid$(buffer, 12, 5))
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(5)
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(6)
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(7)
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(8)
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(9)
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(10)
Line Input #filenumber, buffer      'KONTRL(11)

```

```

Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(12)
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(13)
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(14)
    KONTRL(14) = Val(Mid$(buffer, 13, 5))
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(15)
    KONTRL(15) = Val(Mid$(buffer, 13, 5))
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'KONTRL(16)
Line Input #filenumber, buffer

'read materials (3 lines each)
For i = 1 To NUMMAT
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
Next i

'read nodal data
Line Input #filenumber, buffer      '3 blank lines
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'nodal data header
Line Input #filenumber, buffer
For i = 1 To NNODE + NAXOR
    Line Input #filenumber, buffer
Next i

'read elements
Line Input #filenumber, buffer      'blank line
Line Input #filenumber, buffer      'element header
For i = 1 To NELE
    Line Input #filenumber, buffer
Next i

'read nodal data in different coordinates
Line Input #filenumber, buffer      '3 blank lines
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'nodal header
Line Input #filenumber, buffer
For i = 1 To NNODE
    Line Input #filenumber, buffer
Next i

'read displacement nodes
Line Input #filenumber, buffer      'displacement node header
Line Input #filenumber, buffer
For i = 1 To NUMDIS
    Line Input #filenumber, buffer
Next i

'read primary node mass array
Line Input #filenumber, buffer      'blank line
Line Input #filenumber, buffer      'header
Line Input #filenumber, buffer      'blank line
For i = 1 To NPRI
    Line Input #filenumber, buffer
Next i

'read total length of Q array after ASSBLE
Line Input #filenumber, buffer

'read output code
Line Input #filenumber, buffer      '2 blank lines
Line Input #filenumber, buffer
Line Input #filenumber, buffer      'output header
Line Input #filenumber, buffer      'NPFREQ
    NPFREQ = Val(Mid$(buffer, 18, 10))
Line Input #filenumber, buffer      'NPRU

```

```

    NPRU = Val(Mid$(buffer, 18, 10))
    Line Input #filenumber, buffer    'NPRS
    NPRS = Val(Mid$(buffer, 18, 10))
    Line Input #filenumber, buffer    'NPIC
    NPIC = Val(Mid$(buffer, 18, 10))
    Line Input #filenumber, buffer    'NANG
    Line Input #filenumber, buffer    'NPSEC

    'read nodal disp, vel, accel output
    ReDim ttitles(NPRU)
    Line Input #filenumber, buffer    'header
    For i = 1 To NPRU
        Line Input #filenumber, buffer
        ttitles(i) = Mid$(buffer, 37, 40)
    Next i

    'read selective element stress array output
    If NPRS > 0 Then
        Line Input #filenumber, buffer    'header
        For i = 1 To NPRS
            Line Input #filenumber, buffer
        Next i
    End If

    'read picture output
    If NPIC > 0 Then
        ReDim NPOUT(NPIC)
        Line Input #filenumber, buffer    'header
        Line Input #filenumber, buffer
        For i = 1 To NPIC
            Line Input #filenumber, buffer    '2 blank lines
            Line Input #filenumber, buffer
            Line Input #filenumber, buffer
            NPOUT(i) = CLng(Mid$(buffer, 11, 7))
        Next i
    End If

    'read total length of Q array
    Line Input #filenumber, buffer    'blank line
    Line Input #filenumber, buffer

    'read length of strs array
    Line Input #filenumber, buffer    'blank line
    Line Input #filenumber, buffer

    'read numerical integration data
    Line Input #filenumber, buffer    '4 blank lines
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer    'header
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer    'no. of points
    NPTS = Val(Mid$(buffer, 25, 5))
    Line Input #filenumber, buffer    'initial velocity
    Line Input #filenumber, buffer    'initial disp
    Line Input #filenumber, buffer    'header
    Line Input #filenumber, buffer
    For i = 1 To NPTS
        Line Input #filenumber, buffer
        Line Input #filenumber, buffer    'blank line
    Next i

    'read plane information
    If KONTRL(4) <> 0 Then
        For i = 1 To KONTRL(4)
            Line Input #filenumber, buffer    'blank line
            Line Input #filenumber, buffer    'plane line
            Line Input #filenumber, buffer    'direction cosines
            Line Input #filenumber, buffer    'line 2
            Line Input #filenumber, buffer    'seat back
            Line Input #filenumber, buffer    'associated space nodes
            Line Input #filenumber, buffer    'node numbers
            Line Input #filenumber, buffer    'x, y, z header
            Line Input #filenumber, buffer    'x1
            Line Input #filenumber, buffer
        Next i
    End If

```

```

'x2
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
Next i
End If

'read nodes
If KONTRL(15) <> 0 Then
    Line Input #filenumber, buffer
    JSTART = Val(Mid$(buffer, 1, 10))
    JEND = Val(Mid$(buffer, 11, 10))
    NB = JEND - JSTART
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    For i = 1 To NB
        Line Input #filenumber, buffer
    Next i
End If

'read original element lengths
Line Input #filenumber, buffer
Line Input #filenumber, buffer
Line Input #filenumber, buffer
For i = 1 To NELE
    Line Input #filenumber, buffer
Next i

If NPIC > 0 Then
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    Line Input #filenumber, buffer
    ICMND = Val(Mid$(buffer, 62, 5))
    For i = 1 To ICMND
        Line Input #filenumber, buffer
    Next i
End If

'read output data
numpts = Int(MXSTEP / NPFREQ)
ReDim Plotval(numpts, NPRU + 1)
accum = 0
j = 0
For stepcount = 1 To MXSTEP
    accum = accum + DELT
    If NPIC > 0 Then
        For k = 1 To NPIC
            If stepcount = NPOUT(k) Then
                Line Input #filenumber, buffer
                Line Input #filenumber, buffer
                Line Input #filenumber, buffer
                Line Input #filenumber, buffer
                Line Input #filenumber, buffer
                For i = 1 To ICMND
                    Line Input #filenumber, buffer
                Next i
                For i = 1 To NB + 1
                    Line Input #filenumber, buffer
                Next i
                Line Input #filenumber, buffer
            End If
        Next k
    End If
    If (stepcount Mod NPFREQ) = 0 Then
        j = j + 1
        If KONTRL(4) <> 0 Then
            Line Input #filenumber, buffer
        End If
        Line Input #filenumber, buffer
        If KONTRL(2) = 0 Then
            Plotval(j, 1) = accum
            If NPRU <= 6 Then
                Line Input #filenumber, buffer
            For k = 1 To NPRU

```

```

        Plotval(j, k + 1) = ConvD(Mid$(buffer, 26 + (k - 1) * 16, 16))
    Next k
End If
End If
End If
Next stepcount

```

```

'complex output file format not addressed at this time
'Line Input #filenumber, buffer 'spinal injury likelihood
'Line Input #filenumber, buffer 'max compressive forces
'Line Input #filenumber, buffer 'blank line
'Line Input #filenumber, buffer 'p
'Line Input #filenumber, buffer 'blank line
'Line Input #filenumber, buffer 'bmy
'Line Input #filenumber, buffer 'blank line
'Line Input #filenumber, buffer 'bmz
'Line Input #filenumber, buffer 'blank line
'Line Input #filenumber, buffer 'tp,tmy,tmz
'Line Input #filenumber, buffer 'blank line
'Line Input #filenumber, buffer 'py,bmyy,bmzy
'Line Input #filenumber, buffer 'level
'Line Input #filenumber, buffer 'blank line
'For i = 1 To NB
'    Line Input #filenumber, buffer
'Next i

'call plotting subroutine
Call PlotData(NPRU, numpts)
Close #filenumber

```

End Sub

Sub ReadOutFile ()

```

'subroutine to read output file
'standard output filename is "hsm.out"
' subroutines called: ReadData

Dim buffer As String * 80 'buffer for file record
Dim filenumber As Integer 'file handle

Outfile = App.Path & "\hsm.out"
filenumber = FreeFile
'open file
Open Outfile For Input Access Read As #filenumber Len = 80
On Error GoTo ErrorHandler2
Call ReadData(filenumber, buffer)
Close #filenumber
Exit Sub

ErrorHandler2:
If Err = 62 Then Exit Sub
Resume Next

```

End Sub

**APPENDIX C**  
**INVENTORY OF FILES**

## Archived HSM Files Presented at Kickoff

total 5504

drwxr-xr-x	2	mike	user	5632	Aug 16 14:20	./
drwxrwxrwx	17	root	sys	2560	Jul 21 21:46	./
-rw-r--r--	1	martini	user	6685	Oct 7 1993	ALPHAP.f
-rw-rw-r--	1	martini	user	24244	Oct 7 1993	ALPHAP.o
-rw-r--r--	1	martini	user	7251	Oct 7 1993	ASSBLE.f
-rw-rw-r--	1	martini	user	18680	Oct 7 1993	ASSBLE.o
-rw-r--r--	1	martini	user	62	Oct 7 1993	AXIS.f
-rw-rw-r--	1	martini	user	684	Oct 7 1993	AXIS.o
-rw-r--r--	1	martini	user	3333	Oct 7 1993	BASME.f
-rw-rw-r--	1	martini	user	8792	Oct 7 1993	BASME.o
-rw-r--r--	1	martini	user	8337	Oct 7 1993	BFRGIN.f
-rw-rw-r--	1	martini	user	24372	Oct 7 1993	BFRGIN.o
-rw-r--r--	1	martini	user	54	Oct 7 1993	CARCON.f
-rw-rw-r--	1	martini	user	596	Oct 7 1993	CARCON.o
-rw-r--r--	1	martini	user	706	Oct 7 1993	CROSS.f
-rw-rw-r--	1	martini	user	2104	Oct 7 1993	CROSS.o
-rw-r--r--	1	martini	user	478	Oct 7 1993	DECOD.f
-rw-rw-r--	1	martini	user	1488	Oct 7 1993	DECOD.o
-rw-r--r--	1	martini	user	2309	Oct 7 1993	EIGEN.f
-rw-rw-r--	1	martini	user	8048	Oct 7 1993	EIGEN.o
-rw-r--r--	1	martini	user	64	Oct 7 1993	ELMPLT.f
-rw-rw-r--	1	martini	user	688	Oct 7 1993	ELMPLT.o
-rw-r--r--	1	martini	user	4381	Oct 7 1993	ELOUT.f
-rw-rw-r--	1	martini	user	13080	Oct 7 1993	ELOUT.o
-rw-r--r--	1	martini	user	2970	Oct 7 1993	FRCIN.f
-rw-rw-r--	1	martini	user	8184	Oct 7 1993	FRCIN.o
-rw-rw-r--	1	mike	user	1962	Aug 3 08:00	FREEFD.2
-rw-rw-r--	1	martini	user	1952	Oct 7 1993	FREEFD.f
-rw-rw-r--	1	martini	user	4588	Oct 7 1993	FREEFD.o
-rw-r--r--	1	mike	user	2876	Aug 3 08:00	FREEFD.old
-rw-r--r--	1	martini	user	86	Oct 7 1993	GENFOR.f
-rw-rw-r--	1	martini	user	840	Oct 7 1993	GENFOR.o
-rw-r--r--	1	martini	user	1536	Oct 7 1993	GMPRD.f
-rw-rw-r--	1	martini	user	2068	Oct 7 1993	GMPRD.o
-rw-r--r--	1	martini	user	1516	Oct 7 1993	GTPRD.f
-rw-rw-r--	1	martini	user	2028	Oct 7 1993	GTPRD.o
-rw-r--r--	1	martini	user	955	Oct 7 1993	ICIF.f
-rw-rw-r--	1	martini	user	4544	Oct 7 1993	ICIF.o
-rw-r--r--	1	martini	user	1616	Oct 7 1993	ICIF2.f
-rw-rw-r--	1	martini	user	5452	Oct 7 1993	ICIF2.o
-rw-r--r--	1	martini	user	340	Oct 7 1993	INCODE.f
-rw-rw-r--	1	martini	user	1052	Oct 7 1993	INCODE.o
-rw-r--r--	1	martini	user	58	Oct 7 1993	LINE.f
-rw-rw-r--	1	martini	user	656	Oct 7 1993	LINE.o
-rw-r--r--	1	martini	user	7527	Oct 7 1993	LOCFRG.f
-rw-rw-r--	1	martini	user	19664	Oct 7 1993	LOCFRG.o
-rw-r--r--	1	martini	user	62	Oct 7 1993	LOFIX.f
-rw-rw-r--	1	martini	user	600	Oct 7 1993	LOFIX.o
-rw-r--r--	1	martini	user	11325	Oct 7 1993	OUTPUT.f
-rw-rw-r--	1	martini	user	36696	Oct 7 1993	OUTPUT.o
-rw-r--r--	1	martini	user	56	Oct 7 1993	PEGS.f

-rw-rw-r--	1 jmartini user	640 Oct 7 1993 PEGS.o
-rw-r--r--	1 jmartini user	53 Oct 7 1993 PLOT.f
-rw-rw-r--	1 jmartini user	612 Oct 7 1993 PLOT.o
-rw-r--r--	1 jmartini user	8161 Oct 7 1993 PLOTER.f
-rw-rw-r--	1 jmartini user	13156 Oct 7 1993 PLOTER.o
-rw-r--r--	1 jmartini user	54 Oct 7 1993 PLOTS.f
-rw-rw-r--	1 jmartini user	612 Oct 7 1993 PLOTS.o
-rw-r--r--	1 jmartini user	88 Oct 7 1993 PVFRCN.f
-rw-rw-r--	1 jmartini user	856 Oct 7 1993 PVFRCN.o
-rw-r--r--	1 jmartini user	72 Oct 7 1993 READFD.f
-rw-rw-r--	1 jmartini user	744 Oct 7 1993 READFD.o
-rw-r--r--	1 jmartini user	10852 Oct 7 1993 READIN.f
-rw-rw-r--	1 jmartini user	31860 Oct 7 1993 READIN.o
-rw-r--r--	1 jmartini user	3924 Oct 7 1993 READOU.f
-rw-rw-r--	1 jmartini user	13936 Oct 7 1993 READOU.o
-rw-r--r--	1 jmartini user	64 Oct 7 1993 READXS.f
-rw-rw-r--	1 jmartini user	688 Oct 7 1993 READXS.o
-rw-r--r--	1 jmartini user	47 Oct 7 1993 RELINE.f
-rw-rw-r--	1 jmartini user	564 Oct 7 1993 RELINE.o
-rw-r--r--	1 jmartini user	51 Oct 7 1993 RESTAR.f
-rw-rw-r--	1 jmartini user	580 Oct 7 1993 RESTAR.o
-rw-r--r--	1 jmartini user	847 Oct 7 1993 ROTATE.f
-rw-rw-r--	1 jmartini user	2292 Oct 7 1993 ROTATE.o
-rw-r--r--	1 jmartini user	2002 Oct 7 1993 ROTE.f
-rw-rw-r--	1 jmartini user	6516 Oct 7 1993 ROTE.o
-rw-r--r--	1 jmartini user	55 Oct 7 1993 SCALE.f
-rw-rw-r--	1 jmartini user	628 Oct 7 1993 SCALE.o
-rw-r--r--	1 jmartini user	86 Oct 7 1993 SECACC.f
-rw-rw-r--	1 jmartini user	840 Oct 7 1993 SECACC.o
-rw-r--r--	1 jmartini user	6136 Oct 7 1993 SFRCIN.f
-rw-rw-r--	1 jmartini user	16452 Oct 7 1993 SFRCIN.o
-rw-r--r--	1 jmartini user	6265 Oct 7 1993 SLIDER.f
-rw-rw-r--	1 jmartini user	20092 Oct 7 1993 SLIDER.o
-rw-r--r--	1 jmartini user	10317 Oct 7 1993 SOLVE.f
-rw-rw-r--	1 jmartini user	30808 Oct 7 1993 SOLVE.o
-rw-r--r--	1 jmartini user	13333 Oct 7 1993 SPINIF.f
-rw-rw-r--	1 jmartini user	36340 Oct 7 1993 SPINIF.o
-rw-r--r--	1 jmartini user	60 Oct 7 1993 SYMBOL.f
-rw-rw-r--	1 jmartini user	660 Oct 7 1993 SYMBOL.o
-rw-r--r--	1 jmartini user	77 Oct 7 1993 TASME.f
-rw-rw-r--	1 jmartini user	784 Oct 7 1993 TASME.o
-rw-r--r--	1 jmartini user	94 Oct 7 1993 TFRCIN.f
-rw-rw-r--	1 jmartini user	896 Oct 7 1993 TFRCIN.o
-rw-r--r--	1 jmartini user	67 Oct 7 1993 UASME.f
-rw-rw-r--	1 jmartini user	712 Oct 7 1993 UASME.o
-rw-r--r--	1 jmartini user	70 Oct 7 1993 UFRCIN.f
-rw-rw-r--	1 jmartini user	728 Oct 7 1993 UFRCIN.o
-rw-r--r--	1 jmartini user	6846 Oct 7 1993 UPDATE.f
-rw-rw-r--	1 jmartini user	10672 Oct 7 1993 UPDATE.o
-rw-r--r--	1 jmartini user	492 Oct 7 1993 VECT.f
-rw-rw-r--	1 jmartini user	1748 Oct 7 1993 VECT.o
-rw-r--r--	1 jmartini user	1371 Oct 7 1993 VECTOR.f
-rw-rw-r--	1 jmartini user	3576 Oct 7 1993 VECTOR.o
-rw-r--r--	1 jmartini user	15929 Oct 7 1993 WHAM3.f



```

-rw-rw-r-- 1 jmartini user 34744 Oct 7 1993 WHAM3.o
-rw-r--r-- 1 mike user 30804 Aug 3 08:00 cr24b.inp
-rw-r--r-- 1 mike user 997025 Aug 3 07:59 cr24b.out
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 damp.
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 damp.new
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 damp.out
-rw-rw-r-- 1 root sys 0 Aug 16 14:20 dirhsm
-rw-r--r-- 1 jmartini user 1357 Oct 7 1993 dumyhsm.f
-rw-rw-r-- 1 jmartini user 13896 Oct 7 1993 fort.7
-rw-rw-r-- 1 jmartini user 5150 Oct 7 1993 hb3flx.out
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 hidamp.
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 hidamp.out
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 histif.
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 histif.out
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 hyb2.inp
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 hyb2.out
-rw-r--r-- 1 jmartini user 1467 Oct 7 1993 hyb22.inp
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 hyb22.out
-rw-rw-r-- 1 jmartini user 1798 Oct 7 1993 hyb3fdext.f
-rw-rw-r-- 1 jmartini user 1794 Oct 7 1993 hyb3fdflx.f
-rw-rw-r-- 1 jmartini user 1151 Oct 7 1993 hybIII.flex
-rw-rw-r-- 1 jmartini user 24085 Oct 7 1993 hybIIIflx.out
-rw-rw-r-- 1 jmartini user 544 Jul 7 21:42 info
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 lostif.
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 lostif.out
-rw-r--r-- 1 jmartini user 600 Oct 7 1993 makehsm
-rw-rw-r-- 1 jmartini user 1467 Oct 7 1993 shear.
-rw-rw-r-- 1 jmartini user 60804 Oct 7 1993 shear.out
-rw-rw-r-- 1 jmartini user 44644 Oct 7 1993 shear2.out
-rwxrwxr-x 1 jmartini user 574404 Oct 7 1993 xhsm*

```

# Archived Files in Directory HSM 001

total 106657

```
drwxr-xr-x  2 105  user   4096 Aug 16 14:18 ./
drwxrwxrwx 17 root  sys   2560 Jul 21 21:46 ../
-rw-rw-r--  1 105  user   53784 May 22 1986 ABEPLOT.FTN
-rw-rw-r--  1 105  user   124335 Aug  7 1986 ACES.FTN
-rw-rw-r--  1 105  user    60183 Sep  5 1986 ACES.INP
-rw-rw-r--  1 105  user   1163790 Sep  6 1986 ACES.OUT
-rw-rw-r--  1 105  user    60183 Aug  8 1986 ACES0.INP
-rw-rw-r--  1 105  user   1163790 Aug  8 1986 ACES0.OUT
-rw-rw-r--  1 105  user    60183 Aug  7 1986 ACES1.INP
-rw-rw-r--  1 105  user   1140474 Jun 22 1986 ACES1.OUT
-rw-rw-r--  1 105  user   1163790 Aug 12 1986 ACES2.OUT
-rw-rw-r--  1 105  user    63423 Aug 22 1986 ACES2A.INP
-rw-rw-r--  1 105  user   1439428 Aug 22 1986 ACES2A.OUT
-rw-rw-r--  1 105  user    63423 Aug 22 1986 ACES2B.INP
-rw-rw-r--  1 105  user   1439428 Aug 29 1986 ACES2B.OUT
-rw-rw-r--  1 105  user   1163790 Aug 13 1986 ACES3.OUT
-rw-rw-r--  1 105  user    63423 Aug 22 1986 ACES3A.INP
-rw-rw-r--  1 105  user   1439428 Aug 25 1986 ACES3A.OUT
-rw-rw-r--  1 105  user    63423 Aug 22 1986 ACES3B.INP
-rw-rw-r--  1 105  user   1439428 Sep  2 1986 ACES3B.OUT
-rw-rw-r--  1 105  user    60183 Aug 22 1986 ACES4.INP
-rw-rw-r--  1 105  user   1163790 Aug 13 1986 ACES4.OUT
-rw-rw-r--  1 105  user    63423 Aug 22 1986 ACES4A.INP
-rw-rw-r--  1 105  user   1439428 Aug 25 1986 ACES4A.OUT
-rw-rw-r--  1 105  user    63423 Aug 22 1986 ACES4B.INP
-rw-rw-r--  1 105  user   1439428 Sep  4 1986 ACES4B.OUT
-rw-rw-r--  1 105  user    60183 Aug 22 1986 ACES5.INP
-rw-rw-r--  1 105  user   1163790 Aug 15 1986 ACES5.OUT
-rw-rw-r--  1 105  user    77760 Aug 19 1986 ACESA.FTN
-rw-rw-r--  1 105  user   162243 Apr 21 1986 ACESFD.FTN
-rw-rw-r--  1 105  user    20331 Jul 28 1986 ALPHAP.FTN
-rw-rw-r--  1 105  user     648 Aug 10 1987 BATCH.JEF
-rw-rw-r--  1 105  user     243 Aug  6 1987 BATCH.JJS
-rw-rw-r--  1 105  user    68931 May 17 1985 BOTJEFFD.FTN
-rw-rw-r--  1 105  user   1193940 Jul 24 1986 BUTT.OUT
-rw-rw-r--  1 105  user    50625 Oct  6 1986 CACESFD.FTN
-rw-rw-r--  1 105  user    233766 Aug 11 1986 CBEAM.FTN
-rw-rw-r--  1 105  user    17739 Jan  9 1987 CHECK.FTN
-rw-rw-r--  1 105  user    49977 May 16 1985 CILS.INP
-rw-rw-r--  1 105  user    87237 May 17 1985 CILSFD.FTN
-rw-rw-r--  1 105  user    89667 Sep 18 1986 CMODHARN.FTN
-rw-rw-r--  1 105  user     567 Aug  6 1987 COMBINE.CSS
-rw-rw-r--  1 105  user    16686 Sep  8 1986 COMBINE.FTN
-rw-rw-r--  1 105  user     6834 Aug  6 1987 COMBINE.OUT
-rw-rw-r--  1 105  user   496611 Dec 31 1986 COMBO.FTN
-rw-rw-r--  1 105  user    53460 Oct  6 1986 COMBOFD.FTN
-rw-rw-r--  1 105  user    17658 Jan 12 1987 COMILS.FTN
-rw-rw-r--  1 105  user     243 Oct 15 1986 COMP.CSS
-rw-rw-r--  1 105  user    2673 Apr 22 1986 DRIACES.INP
-rw-rw-r--  1 105  user    3483 Aug  7 1984 DRIVROD.FTN
-rw-rw-r--  1 105  user    3402 Jun  6 1985 EIGEN.FTN
```

-rw-rw-r--	1	105	user	11514	Jun 13	1987	F7CETRAP.CSS
-rw-rw-r--	1	105	user	162	Oct 6	1986	FAST.CSS
-rw-rw-r--	1	105	user	11097	Feb 28	1986	FDSSMCS.FTN
-rw-rw-r--	1	105	user	75978	May 23	1986	FHHENC.FTN
-rw-rw-r--	1	105	user	4293	Feb 27	1986	FIVEDOF.INP
-rw-rw-r--	1	105	user	3564	Apr 22	1986	FREEFDAC.FTN
-rw-rw-r--	1	105	user	5832	Feb 27	1986	FREEFDDO.FTN
-rw-rw-r--	1	105	user	15471	Jan 8	1987	FU.FTN
-rw-rw-r--	1	105	user	729	Jul 16	1986	GETIT.FTN
-rw-rw-r--	1	105	user	76545	Jun 25	1986	H7.FTN
-rw-rw-r--	1	105	user	59292	Jul 2	1986	H7.INP
-rw-rw-r--	1	105	user	1016390	Sep 6	1986	H7.OUT
-rw-rw-r--	1	105	user	122958	Sep 5	1986	H7A.FTN
-rw-rw-r--	1	105	user	124011	Sep 10	1986	HACESFD.FTN
-rw-rw-r--	1	105	user	39771	Feb 27	1986	HCS685.FTN
-rw-rw-r--	1	105	user	508113	May 16	1986	HCSCODE.FTN
-rw-rw-r--	1	105	user	59292	Aug 6	1987	HCSILS.INP
-rw-rw-r--	1	105	user	1254776	Aug 10	1987	HCSILS.OUT
-rw-rw-r--	1	105	user	12150	Aug 10	1987	HCSILS.PUN
-rw-rw-r--	1	105	user	91773	Jun 18	1986	HCSILSFD.FTN
-rw-rw-r--	1	105	user	96309	Aug 4	1987	HCSM.INP
-rw-rw-r--	1	105	user	2420442	Aug 6	1987	HCSM.OUT
-rw-rw-r--	1	105	user	2187	Aug 7	1987	HCSM.PUN
-rw-rw-r--	1	105	user	474984	Jun 24	1985	HCSMAR.FTN
-rw-rw-r--	1	105	user	472797	Jun 27	1985	HCSMARJ.FTN
-rw-rw-r--	1	105	user	472878	Feb 14	1986	HCSMARM.FTN
-rw-rw-r--	1	105	user	472878	Jan 28	1986	HCSMART.FTN
-rw-rw-r--	1	105	user	473445	May 20	1985	HCSMDLT.FTN
-rw-rw-r--	1	105	user	473931	Jun 11	1985	HCSMDTT.FTN
-rw-rw-r--	1	105	user	127656	Oct 1	1986	HCSMFD.FTN
-rw-rw-r--	1	105	user	548462	Aug 5	1987	HCSMFD.LST
-rw-rw-r--	1	105	user	465426	May 22	1985	HCSMRS2.FTN
-rw-rw-r--	1	105	user	465669	Jul 26	1985	HCSMRS3.FTN
-rw-rw-r--	1	105	user	479682	Apr 25	1985	HCSMRST.FTN
-rw-rw-r--	1	105	user	124416	Aug 7	1986	HHENC.FTN
-rw-rw-r--	1	105	user	60021	Aug 7	1986	HHENC.INP
-rw-rw-r--	1	105	user	1153874	Aug 8	1986	HHENC.OUT
-rw-rw-r--	1	105	user	60021	Aug 7	1986	HHENC1.INP
-rw-rw-r--	1	105	user	1153874	Jul 25	1986	HHENC1.OUT
-rw-rw-r--	1	105	user	124173	Jul 24	1986	HHENC2.FTN
-rw-rw-r--	1	105	user	60507	Jul 24	1986	HHENC2.INP
-rw-rw-r--	1	105	user	1200506	Jul 24	1986	HHENC2.OUT
-rw-rw-r--	1	105	user	49410	Aug 6	1987	HILS.INP
-rw-rw-r--	1	105	user	73953	May 14	1985	HILSFD.FTN
-rw-rw-r--	1	105	user	20331	Apr 5	1985	HSALPHAP.FTN
-rw-rw-r--	1	105	user	378351	Apr 24	1987	HSM.FTN
-rw-rw-r--	1	105	user	11988	Dec 5	1984	HSSSM.FTN
-rw-rw-r--	1	105	user	496611	Aug 4	1987	HTSM.FTN
-rw-rw-r--	1	105	user	151065	Aug 4	1987	HTSM.INP
-rw-rw-r--	1	105	user	77193	Aug 4	1987	HTSMFD.FTN
-rw-rw-r--	1	105	user	306860	Aug 10	1987	HTSMFD.LST
-rw-rw-r--	1	105	user	91287	Aug 4	1987	ILSFD.FTN
-rw-rw-r--	1	105	user	123280	Aug 10	1987	ILSFD.LST
-rw-rw-r--	1	105	user	59697	Jul 27	1987	ILSFRC.INP

-rw-rw-r--	1 105	user	60183 May 14 1987 ILSRET.INP
-rw-rw-r--	1 105	user	741824 May 15 1987 ILSRET.OUT
-rw-rw-r--	1 105	user	14256 May 15 1987 ILSRET.PUN
-rw-rw-r--	1 105	user	2592 Oct 22 1985 ITERATE.FTN
-rw-rw-r--	1 105	user	138510 Jan 21 1987 J.INP
-rw-rw-r--	1 105	user	567490 Jan 12 1987 J.OUT
-rw-rw-r--	1 105	user	60183 Aug 22 1986 JACES.INP
-rw-rw-r--	1 105	user	1203990 Sep 8 1986 JACES.OUT
-rw-rw-r--	1 105	user	124497 Jul 29 1986 JEF.FTN
-rw-rw-r--	1 105	user	30132 May 17 1985 JEFFFD.FTN
-rw-rw-r--	1 105	user	12231 May 22 1985 JSHCSSSM.INP
-rw-rw-r--	1 105	user	74277 May 16 1985 JSILSFD.FTN
-rw-rw-r--	1 105	user	12717 Jul 25 1986 JSSM.INP
-rw-rw-r--	1 105	user	648 Oct 17 1986 LIG.FTN
-rw-rw-r--	1 105	user	6318 Nov 30 1987 LINKHCS.CSS
-rw-rw-r--	1 105	user	20979 Jul 6 1984 LOCFRC.FTN
-rw-rw-r--	1 105	user	140940 Jan 9 1987 LONGB.INP
-rw-rw-r--	1 105	user	0 Jan 9 1987 LONGB.OUT
-rw-rw-r--	1 105	user	2268 Aug 21 1985 MULTIPLY.FTN
-rw-rw-r--	1 105	user	536 Jan 8 1987 MULTIPLY.OUT
-rw-rw-r--	1 105	user	46332 Apr 29 1985 NECKSSM.FTN
-rw-rw-r--	1 105	user	76626 May 24 1985 NKACBOT.FTN
-rw-rw-r--	1 105	user	1701 Feb 28 1986 ONEDOF.INP
-rw-rw-r--	1 105	user	810 Oct 15 1986 PLOTIG.FTN
-rw-rw-r--	1 105	user	7938 Jul 22 1986 PLOTMULT.FTN
-rw-rw-r--	1 105	user	4455 Mar 27 1986 PLOTR.FTN
-rw-rw-r--	1 105	user	6075 Sep 12 1986 PLOTR2.FTN
-rw-rw-r--	1 105	user	8910 Nov 17 1986 PLOTSIF.FTN
-rw-rw-r--	1 105	user	5832 Jul 8 1985 PLOTTER.FTN
-rw-rw-r--	1 105	user	6642 Jun 19 1985 PLOTTER2.FTN
-rw-rw-r--	1 105	user	27540 Jul 24 1984 POLYFIT.FTN
-rw-rw-r--	1 105	user	567 Nov 5 1986 READIT.FTN
-rw-rw-r--	1 105	user	88452 Mar 6 1986 REC50FD.FTN
-rw-rw-r--	1 105	user	53460 Oct 6 1986 RESTARFD.FTN
-rw-rw-r--	1 105	user	24786 Mar 11 1987 RESTART.FTN
-rw-rw-r--	1 105	user	810 May 13 1987 RESTART.INP
-rw-rw-r--	1 105	user	109026 Aug 10 1987 RETRACT.FTN
-rw-rw-r--	1 105	user	50706 Aug 10 1987 RETRACT.INP
-rw-rw-r--	1 105	user	5994 Aug 7 1984 ROD.INP
-rw-rw-r--	1 105	user	494019 Oct 2 1986 RSSMCS.FTN
-rw-rw-r--	1 105	user	8829 Jul 15 1986 RUNFD.FTN
-rw-rw-r--	1 105	user	9639 Mar 22 1987 RUNHCS.CSS
-rw-rw-r--	1 105	user	10044 Apr 19 1985 RUNHCSA.CSS
-rw-rw-r--	1 105	user	9963 Jun 14 1987 RUNHCSAR.CSS
-rw-rw-r--	1 105	user	10044 Sep 18 1987 RUNHCSAT.CSS
-rw-rw-r--	1 105	user	9963 May 13 1987 RUNHCSCC.CSS
-rw-rw-r--	1 105	user	10125 Sep 3 1986 RUNHCSM.CSS
-rw-rw-r--	1 105	user	6561 Mar 4 1987 RUNHCSO.CSS
-rw-rw-r--	1 105	user	9963 Apr 24 1987 RUNHCSR.CSS
-rw-rw-r--	1 105	user	9963 May 8 1987 RUNHCSR2.CSS
-rw-rw-r--	1 105	user	9963 Oct 11 1985 RUNHCSR3.CSS
-rw-rw-r--	1 105	user	10125 Nov 17 1986 RUNHCSRJ.CSS
-rw-rw-r--	1 105	user	9963 May 20 1987 RUNHCST.CSS
-rw-rw-r--	1 105	user	9963 May 31 1987 RUNHCSTT.CSS

-rw-rw-r--	1 105	user	10125 Aug 4 1987	RUNHTSM.CSS
-rw-rw-r--	1 105	user	10044 Dec 24 1986	RUNSSMS.CSS
-rw-rw-r--	1 105	user	10044 Sep 18 1987	RUNTEMP3.CSS
-rw-rw-r--	1 105	user	8910 Nov 17 1986	SAFE.FTN
-rw-rw-r--	1 105	user	140940 Oct 6 1986	SIM0.INP
-rw-rw-r--	1 105	user	151065 Jan 6 1987	SIM1.INP
-rw-rw-r--	1 105	user	796630 Dec 29 1986	SIM1.OUT
-rw-rw-r--	1 105	user	151065 Dec 29 1986	SIM1ANTM.INP
-rw-rw-r--	1 105	user	3052520 Dec 30 1986	SIM1ANTM.OUT
-rw-rw-r--	1 105	user	77274 Dec 23 1986	SIM1FD.FTN
-rw-rw-r--	1 105	user	140940 Oct 31 1986	SIM2.INP
-rw-rw-r--	1 105	user	567490 Dec 23 1986	SIM2.OUT
-rw-rw-r--	1 105	user	116964 Oct 31 1986	SIM2BFD.FTN
-rw-rw-r--	1 105	user	116964 Oct 31 1986	SIM2FD.FTN
-rw-rw-r--	1 105	user	140940 Oct 6 1986	SIM3.INP
-rw-rw-r--	1 105	user	70389 Oct 6 1986	SIM3FD.FTN
-rw-rw-r--	1 105	user	30537 Oct 2 1986	SIMFD.FTN
-rw-rw-r--	1 105	user	6399 Dec 31 1984	SINFD.FTN
-rw-rw-r--	1 105	user	27297 Sep 3 1986	SINFDA.FTN
-rw-rw-r--	1 105	user	98334 Oct 15 1985	SLED.INP
-rw-rw-r--	1 105	user	160704 Oct 30 1985	SLEDFD.FTN
-rw-rw-r--	1 105	user	15471 Jul 6 1984	SLIDER.FTN
-rw-rw-r--	1 105	user	15552 Jul 6 1984	SLIDERA.FTN
-rw-rw-r--	1 105	user	60183 Jan 13 1987	SPEC1.INP
-rw-rw-r--	1 105	user	165222 Jan 13 1987	SPEC1.OUT
-rw-rw-r--	1 105	user	60183 Jan 13 1987	SPEC2.INP
-rw-rw-r--	1 105	user	1163790 Jan 6 1987	SPEC2.OUT
-rw-rw-r--	1 105	user	60183 Jan 13 1987	SPEC3.INP
-rw-rw-r--	1 105	user	1163790 Jan 8 1987	SPEC3.OUT
-rw-rw-r--	1 105	user	60183 Jan 13 1987	SPEC4.INP
-rw-rw-r--	1 105	user	1163790 Jan 9 1987	SPEC4.OUT
-rw-rw-r--	1 105	user	60183 Jan 5 1987	SPEC5.INP
-rw-rw-r--	1 105	user	164552 Jan 9 1987	SPEC5.OUT
-rw-rw-r--	1 105	user	60183 Jan 5 1987	SPEC6.INP
-rw-rw-r--	1 105	user	12717 Aug 6 1987	SSM.INP
-rw-rw-r--	1 105	user	451848 Aug 6 1987	SSM.OUT
-rw-rw-r--	1 105	user	13689 Aug 6 1987	SSM.PUN
-rw-rw-r--	1 105	user	5751 Apr 18 1985	SSM1DOF.FTN
-rw-rw-r--	1 105	user	5508 May 2 1985	SSM1DOF1.FTN
-rw-rw-r--	1 105	user	1620 May 20 1985	SSM1DOF1.INP
-rw-rw-r--	1 105	user	2106 May 31 1985	SSM2DOF.INP
-rw-rw-r--	1 105	user	2106 Jun 6 1985	SSM2DOFA.INP
-rw-rw-r--	1 105	user	99387 Sep 11 1986	SSMACES.INP
-rw-rw-r--	1 105	user	53865 Aug 4 1987	SSMFD.FTN
-rw-rw-r--	1 105	user	257950 Aug 6 1987	SSMFD.LST
-rw-rw-r--	1 105	user	8181 May 24 1985	SSMGZ4.FTN
-rw-rw-r--	1 105	user	18549 May 16 1985	SSMPLOT.FTN
-rw-rw-r--	1 105	user	5670 Feb 12 1985	SSMSTEP.FTN
-rw-rw-r--	1 105	user	87885 May 15 1986	STANDARD.INP
-rw-rw-r--	1 105	user	10287 Dec 19 1986	STEP.CSS
-rw-rw-r--	1 105	user	43092 Dec 19 1986	STEP.FTN
-rw-rw-r--	1 105	user	147906 Dec 23 1986	STEP.INP
-rw-rw-r--	1 105	user	578344 Dec 23 1986	STEP.OUT
-rw-rw-r--	1 105	user	96309 Jul 12 1985	STRESS.INP

-rw-rw-r--	1 105	user	18468 Aug 20 1986 SUBTN.FTN
-rw-rw-r--	1 105	user	16848 Aug 20 1986 SUBTRACT.FTN
-rw-rw-r--	1 105	user	6834 Aug 20 1986 SUBTRACT.OUT
-rw-rw-r--	1 105	user	1215 Aug 18 1986 SWITCH.FTN
-rw-rw-r--	1 105	user	810 Dec 7 1986 TEMP.FTN
-rw-rw-r--	1 105	user	7533 Feb 28 1986 TENDOF.INP
-rw-rw-r--	1 105	user	230300 May 7 1970 TESTREP.LAC
-rw-rw-r--	1 105	user	2997 Feb 28 1986 THREEDOF.INP
-rw-rw-r--	1 105	user	255672 Feb 28 1986 THREEDOF.OUT
-rw-rw-r--	1 105	user	2673 May 9 1985 TRAP.FTN
-rw-rw-r--	1 105	user	2916 Jun 18 1985 TRAPT.FTN
-rw-rw-r--	1 105	user	324 Oct 2 1986 TRIGGER.FTN
-rw-rw-r--	1 105	user	2592 May 9 1985 TRYTRAP.FTN
-rw-rw-r--	1 105	user	2349 Feb 28 1986 TWODOF.INP
-rw-rw-r--	1 105	user	216142 Feb 28 1986 TWODOF.OUT
-rw-rw-r--	1 105	user	77586 Aug 27 1986 VISCOID.FTN
-rw-rw-r--	1 105	user	35721 Aug 10 1987 WHAM3.FTN
-rw-rw-r--	1 105	user	42120 May 15 1985 WHAMA.FTN
-rw-rw-r--	1 105	user	42444 May 20 1985 WHAMDLT.FTN
-rw-rw-r--	1 105	user	42363 Nov 26 1984 WHAMII.FTN
-rw-rw-r--	1 105	user	42444 Sep 25 1985 WHAMRS2.FTN
-rw-rw-r--	1 105	user	42930 Feb 4 1987 WHAMRSJ.FTN
-rw-rw-r--	1 105	user	42363 Mar 11 1985 WHAMSM.FTN
-rw-rw-r--	1 105	user	42768 Feb 26 1986 WHAMSSM.FTN
-rw-rw-r--	1 root	sys	0 Aug 16 14:18 dir001

## Archived Files in Directory HSM 002

total 2191

drwxr-xr-x	2	105	user	8192 Aug 16 14:19 ./
drwxrwxrwx	17	root	sys	2560 Jul 21 21:46 ../
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL1ACC.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL1VEL.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL3VEL.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL5ACC.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL5VEL.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL7ACC.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL7VEL.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL9ACC.INP
-rw-rw-r--	1	105	user	486 Feb 25 1985 EL9VEL.INP
-rw-rw-r--	1	105	user	3078 Dec 14 1984 FRCRILS.FTN
-rw-rw-r--	1	105	user	3321 Dec 7 1984 FRCRTC.FTN
-rw-rw-r--	1	105	user	3321 Jan 29 1985 FRCRTC20.FTN
-rw-rw-r--	1	105	user	3402 Jan 9 1985 FRCRTC2D.FTN
-rw-rw-r--	1	105	user	3564 Jun 24 1985 FREAS1.FTN
-rw-rw-r--	1	105	user	3564 Jul 10 1985 FREAS10.FTN
-rw-rw-r--	1	105	user	3564 Jul 10 1985 FREAS11.FTN
-rw-rw-r--	1	105	user	3564 Jun 25 1985 FREAS2.FTN
-rw-rw-r--	1	105	user	3564 Jun 26 1985 FREAS2B.FTN
-rw-rw-r--	1	105	user	3564 Jun 25 1985 FREAS3.FTN
-rw-rw-r--	1	105	user	3564 Jun 25 1985 FREAS4.FTN
-rw-rw-r--	1	105	user	3564 Jun 26 1985 FREAS4B.FTN
-rw-rw-r--	1	105	user	3564 Jun 27 1985 FREAS5.FTN
-rw-rw-r--	1	105	user	3564 Jun 28 1985 FREAS6.FTN
-rw-rw-r--	1	105	user	3564 Jul 1 1985 FREAS7.FTN
-rw-rw-r--	1	105	user	3564 Jul 3 1985 FREAS8.FTN
-rw-rw-r--	1	105	user	3564 Jul 3 1985 FREAS9.FTN
-rw-rw-r--	1	105	user	3564 Jul 11 1985 FREASL.FTN
-rw-rw-r--	1	105	user	486 Oct 15 1984 FRFDSIN.FTN
-rw-rw-r--	1	105	user	3564 Jun 21 1985 FRFRC1.FTN
-rw-rw-r--	1	105	user	3564 Jun 21 1985 FRFRC10.FTN
-rw-rw-r--	1	105	user	1539 Nov 27 1984 FRSK2.FTN
-rw-rw-r--	1	105	user	3564 Jun 24 1985 FRSK207.FTN
-rw-rw-r--	1	105	user	3564 Jun 14 1985 FRSK20H.FTN
-rw-rw-r--	1	105	user	3564 Jun 18 1985 FRSK20I.FTN
-rw-rw-r--	1	105	user	3564 Jun 18 1985 FRSK20J.FTN
-rw-rw-r--	1	105	user	3564 Jun 20 1985 FRSK20K.FTN
-rw-rw-r--	1	105	user	4617 Jun 24 1985 FRSK20L.FTN
-rw-rw-r--	1	105	user	4617 Jun 21 1985 FRSK20M.FTN
-rw-rw-r--	1	105	user	3564 Jun 4 1985 FRSK23.FTN
-rw-rw-r--	1	105	user	3564 Jun 21 1985 FRSK236.FTN
-rw-rw-r--	1	105	user	3564 Jun 21 1985 FRSK237.FTN
-rw-rw-r--	1	105	user	3564 Jun 21 1985 FRSK238.FTN
-rw-rw-r--	1	105	user	3564 Jun 5 1985 FRSK23B.FTN
-rw-rw-r--	1	105	user	3564 Jun 6 1985 FRSK23C.FTN
-rw-rw-r--	1	105	user	3564 Jun 11 1985 FRSK23D.FTN
-rw-rw-r--	1	105	user	3564 Jun 11 1985 FRSK23E.FTN
-rw-rw-r--	1	105	user	3564 Jun 12 1985 FRSK23F.FTN
-rw-rw-r--	1	105	user	3564 Jun 14 1985 FRSK23H.FTN
-rw-rw-r--	1	105	user	3564 Jun 4 1985 FRSK24.FTN

-rw-rw-r--	1 105	user	3564 Jun 12 1985 FRSK24F.FTN
-rw-rw-r--	1 105	user	3564 Jun 13 1985 FRSK24G.FTN
-rw-rw-r--	1 105	user	3564 Jun 4 1985 FRSK25.FTN
-rw-rw-r--	1 105	user	3564 Jun 21 1985 FRSK256.FTN
-rw-rw-r--	1 105	user	3564 Jun 21 1985 FRSK257.FTN
-rw-rw-r--	1 105	user	3564 Jun 21 1985 FRSK258.FTN
-rw-rw-r--	1 105	user	3564 Jun 14 1985 FRSK25H.FTN
-rw-rw-r--	1 105	user	3564 Jun 4 1985 FRSK26.FTN
-rw-rw-r--	1 105	user	3564 Jun 21 1985 FRSK267.FTN
-rw-rw-r--	1 105	user	3564 Jun 12 1985 FRSK26F.FTN
-rw-rw-r--	1 105	user	3564 Jun 14 1985 FRSK27H.FTN
-rw-rw-r--	1 105	user	3564 Jun 12 1985 FRSK28F.FTN
-rw-rw-r--	1 105	user	3078 Jan 9 1985 FRSK2A.FTN
-rw-rw-r--	1 105	user	3078 Jan 23 1985 FRSK2B.FTN
-rw-rw-r--	1 105	user	3078 May 1 1985 FRSK2C.FTN
-rw-rw-r--	1 105	user	3564 Jun 4 1985 FRSK30.FTN
-rw-rw-r--	1 105	user	3564 Jun 12 1985 FRSK30F.FTN
-rw-rw-r--	1 105	user	3564 Jun 14 1985 FRSK30H.FTN
-rw-rw-r--	1 105	user	3078 Jun 5 1985 FRSKFRC.FTN
-rw-rw-r--	1 105	user	3483 Jun 14 1985 FRSKFRC1.FTN
-rw-rw-r--	1 105	user	3483 Jun 14 1985 FRSKFRC2.FTN
-rw-rw-r--	1 105	user	4617 Jun 20 1985 FRSKFRC3.001
-rw-rw-r--	1 105	user	4617 Jun 21 1985 FRSKFRC3.FTN
-rw-rw-r--	1 105	user	4617 Jun 21 1985 FRSKFRC4.FTN
-rw-rw-r--	1 105	user	3078 Jun 6 1985 FRSKFRCB.FTN
-rw-rw-r--	1 105	user	4131 May 24 1985 HCS1ASA.INP
-rw-rw-r--	1 105	user	486 May 28 1985 HCS1ASA.PUN
-rw-rw-r--	1 105	user	4131 May 20 1985 HCS1CGA.INP
-rw-rw-r--	1 105	user	486 May 21 1985 HCS1CGA.PUN
-rw-rw-r--	1 105	user	4131 May 20 1985 HCS1CGP.INP
-rw-rw-r--	1 105	user	486 May 21 1985 HCS1CGP.PUN
-rw-rw-r--	1 105	user	486 May 24 1985 HCS20G40.FTN
-rw-rw-r--	1 105	user	4131 Jun 3 1985 HCS2CGA.INP
-rw-rw-r--	1 105	user	4131 May 21 1985 HCS2CGP.INP
-rw-rw-r--	1 105	user	486 May 23 1985 HCS2CGP.PUN
-rw-rw-r--	1 105	user	4131 May 19 1985 HCS3CGA.INP
-rw-rw-r--	1 105	user	486 May 21 1985 HCS3CGA.PUN
-rw-rw-r--	1 105	user	4131 May 21 1985 HCS3CGP.INP
-rw-rw-r--	1 105	user	486 May 24 1985 HCS3CGP.PUN
-rw-rw-r--	1 105	user	2025 Jul 1 1985 HCS685.FTN
-rw-rw-r--	1 105	user	4131 Sep 25 1985 HCS685A1.INP
-rw-rw-r--	1 105	user	486 Sep 26 1985 HCS685A1.PUN
-rw-rw-r--	1 105	user	972 Sep 28 1985 HCS685A2.PUN
-rw-rw-r--	1 105	user	972 Oct 1 1985 HCS685A3.PUN
-rw-rw-r--	1 105	user	486 Oct 11 1985 HCS685A4.PUN
-rw-rw-r--	1 105	user	486 Oct 3 1985 HCS685A5.PUN
-rw-rw-r--	1 105	user	486 Oct 11 1985 HCS685R1.INP
-rw-rw-r--	1 105	user	486 Oct 12 1985 HCS685R1.PUN
-rw-rw-r--	1 105	user	486 Oct 11 1985 HCS685R2.INP
-rw-rw-r--	1 105	user	486 Oct 13 1985 HCS685R2.PUN
-rw-rw-r--	1 105	user	4131 Mar 22 1985 HCSGZ1KA.INP
-rw-rw-r--	1 105	user	486 Mar 22 1985 HCSGZ1KA.PUN
-rw-rw-r--	1 105	user	486 Mar 26 1985 HCSGZ1KB.PUN
-rw-rw-r--	1 105	user	4131 Apr 4 1985 HCSGZ1KG.INP



-rw-rw-r--	1 105	user	486 Mar 28 1985 HCSGZ1KG.PUN
-rw-rw-r--	1 105	user	4131 Apr 1 1985 HCSGZ1KH.INP
-rw-rw-r--	1 105	user	486 Apr 2 1985 HCSGZ1KH.PUN
-rw-rw-r--	1 105	user	4050 Jan 28 1985 HCSGZ1SP.INP
-rw-rw-r--	1 105	user	4131 Mar 8 1985 HCSGZ2KB.INP
-rw-rw-r--	1 105	user	486 Mar 15 1985 HCSGZ2KB.PUN
-rw-rw-r--	1 105	user	4131 Mar 11 1985 HCSGZ2KC.INP
-rw-rw-r--	1 105	user	486 Mar 15 1985 HCSGZ2KC.PUN
-rw-rw-r--	1 105	user	4131 Mar 13 1985 HCSGZ2KD.INP
-rw-rw-r--	1 105	user	486 Mar 18 1985 HCSGZ2KD.PUN
-rw-rw-r--	1 105	user	4131 Mar 14 1985 HCSGZ2KE.INP
-rw-rw-r--	1 105	user	486 Mar 18 1985 HCSGZ2KE.PUN
-rw-rw-r--	1 105	user	4131 Mar 15 1985 HCSGZ2KF.INP
-rw-rw-r--	1 105	user	486 Mar 20 1985 HCSGZ2KF.PUN
-rw-rw-r--	1 105	user	4131 Mar 22 1985 HCSGZ2KG.INP
-rw-rw-r--	1 105	user	486 Mar 25 1985 HCSGZ2KG.PUN
-rw-rw-r--	1 105	user	4050 Dec 13 1984 HCSGZ2SP.INP
-rw-rw-r--	1 105	user	4131 Mar 22 1985 HCSGZ3KA.INP
-rw-rw-r--	1 105	user	486 Mar 26 1985 HCSGZ3KA.PUN
-rw-rw-r--	1 105	user	4131 Jan 18 1985 HCSGZ3SP.INP
-rw-rw-r--	1 105	user	4131 Mar 6 1985 HCSGZ42K.INP
-rw-rw-r--	1 105	user	486 May 21 1985 HCSGZ42K.PUN
-rw-rw-r--	1 105	user	4131 Apr 29 1985 HCSGZ4R.FTN
-rw-rw-r--	1 105	user	4131 May 24 1985 HCSGZ4SP.INP
-rw-rw-r--	1 105	user	486 May 28 1985 HCSGZ4SP.PUN
-rw-rw-r--	1 105	user	4131 May 24 1985 HCSGZ5SP.INP
-rw-rw-r--	1 105	user	486 May 17 1985 HCSGZ5SP.PUN
-rw-rw-r--	1 105	user	486 May 13 1985 HCSMCC.PUN
-rw-rw-r--	1 105	user	20736 Mar 11 1985 HCSMREF.FTN
-rw-rw-r--	1 105	user	20736 May 7 1985 HCSMRST.FTN
-rw-rw-r--	1 105	user	21222 May 7 1985 HCSMRSTO.FTN
-rw-rw-r--	1 105	user	486 Apr 8 1985 HCSRSTA1.PUN
-rw-rw-r--	1 105	user	486 Apr 8 1985 HCSRSTA2.PUN
-rw-rw-r--	1 105	user	486 Apr 8 1985 HCSRSTB1.PUN
-rw-rw-r--	1 105	user	972 Apr 8 1985 HCSRSTB2.PUN
-rw-rw-r--	1 105	user	486 Jun 12 1985 HCSSTA1R.PUN
-rw-rw-r--	1 105	user	486 Jun 12 1985 HCSSTA2R.PUN
-rw-rw-r--	1 105	user	162 Jul 29 1985 HCSSTA3R.INP
-rw-rw-r--	1 105	user	486 Jul 30 1985 HCSSTA3R.PUN
-rw-rw-r--	1 105	user	4131 Jun 3 1985 HCSSTDA.INP
-rw-rw-r--	1 105	user	486 Jun 3 1985 HCSSTDA.PUN
-rw-rw-r--	1 105	user	4131 Jun 4 1985 HCSSTDA1.INP
-rw-rw-r--	1 105	user	486 Jun 12 1985 HCSSTDA1.PUN
-rw-rw-r--	1 105	user	4131 Jun 7 1985 HCSSTDA2.INP
-rw-rw-r--	1 105	user	486 Jun 12 1985 HCSSTDA2.PUN
-rw-rw-r--	1 105	user	2592 Jul 26 1985 HCSSTDA3.FTN
-rw-rw-r--	1 105	user	4131 Jul 26 1985 HCSSTDA3.INP
-rw-rw-r--	1 105	user	486 Jul 29 1985 HCSSTDA3.PUN
-rw-rw-r--	1 105	user	4131 Jun 18 1985 HCSTEST.INP
-rw-rw-r--	1 105	user	486 Jun 19 1985 HCSTEST.PUN
-rw-rw-r--	1 105	user	4131 Jun 14 1985 HCSTEST1.INP
-rw-rw-r--	1 105	user	486 Jun 16 1985 HCSTEST1.PUN
-rw-rw-r--	1 105	user	243 Jun 19 1985 HCSTESTR.INP
-rw-rw-r--	1 105	user	486 Jun 19 1985 HCSTESTR.PUN

-rw-rw-r--	1 105	user	486 Jun 19 1985 HCSTSTR2.INP
-rw-rw-r--	1 105	user	486 Jun 25 1985 HCSTSTR2.PUN
-rw-rw-r--	1 105	user	2025 Jan 17 1985 HS3SK1B.INP
-rw-rw-r--	1 105	user	2025 Apr 26 1985 HS3SK1C.INP
-rw-rw-r--	1 105	user	2025 Jan 21 1985 HS3SK3B.INP
-rw-rw-r--	1 105	user	2025 Jan 23 1985 HS3SK5B.INP
-rw-rw-r--	1 105	user	2025 Apr 29 1985 HS3SK5C.INP
-rw-rw-r--	1 105	user	2025 Jan 22 1985 HS3SK7B.INP
-rw-rw-r--	1 105	user	2025 Apr 29 1985 HS3SK7C.INP
-rw-rw-r--	1 105	user	486 Mar 12 1985 JADJUST.FTN
-rw-rw-r--	1 105	user	1539 Mar 12 1985 JBABALPH.FTN
-rw-rw-r--	1 105	user	486 Mar 12 1985 JBABFREE.FTN
-rw-rw-r--	1 105	user	1539 Mar 12 1985 JBABOON.DTA
-rw-rw-r--	1 105	user	486 Mar 12 1985 JBOONTST.CSS
-rw-rw-r--	1 105	user	9581 Mar 12 1985 JC.
-rw-rw-r--	1 105	user	505 Mar 12 1985 JCODETST.CSS
-rw-rw-r--	1 105	user	324 Mar 12 1985 JDOFTEST.DTA
-rw-rw-r--	1 105	user	486 Mar 12 1985 JDOFTEST.FTN
-rw-rw-r--	1 105	user	486 Mar 12 1985 JDRECTRY.
-rw-rw-r--	1 105	user	486 Mar 12 1985 KFLEX1.AIN
-rw-rw-r--	1 105	user	486 Mar 12 1985 KFLEX2.AIN
-rw-rw-r--	1 105	user	1539 Mar 12 1985 KHS.FTN
-rw-rw-r--	1 105	user	486 Mar 12 1985 KHS1.INP
-rw-rw-r--	1 105	user	486 Mar 12 1985 KHS2.INP
-rw-rw-r--	1 105	user	486 Mar 12 1985 KHS3.INP
-rw-rw-r--	1 105	user	486 Mar 12 1985 KHS4.INP
-rw-rw-r--	1 105	user	402 Mar 12 1985 KLIST.CO
-rw-rw-r--	1 105	user	402 Mar 12 1985 KOUTPUT.1
-rw-rw-r--	1 105	user	13400 Mar 12 1985 KOUTPUT.AOU
-rw-rw-r--	1 105	user	486 Mar 12 1985 KVIEW.VIN
-rw-rw-r--	1 105	user	1974 Mar 12 1985 KVIEW.VPL
-rw-rw-r--	1 105	user	2592 Mar 26 1985 LDEL0.INP
-rw-rw-r--	1 105	user	486 Mar 27 1985 LDEL0.PUN
-rw-rw-r--	1 105	user	2592 Mar 26 1985 LDEL0R.INP
-rw-rw-r--	1 105	user	486 Mar 27 1985 LDEL0R.PUN
-rw-rw-r--	1 105	user	972 Mar 26 1985 LDEL0ROT.INP
-rw-rw-r--	1 105	user	2592 Mar 27 1985 LDEL10.INP
-rw-rw-r--	1 105	user	486 Mar 28 1985 LDEL10.PUN
-rw-rw-r--	1 105	user	2592 Mar 27 1985 LDEL15.INP
-rw-rw-r--	1 105	user	486 Mar 28 1985 LDEL15.PUN
-rw-rw-r--	1 105	user	2592 Mar 27 1985 LDEL20.INP
-rw-rw-r--	1 105	user	486 Mar 28 1985 LDEL20.PUN
-rw-rw-r--	1 105	user	486 Mar 23 1985 LDEL20A.PUN
-rw-rw-r--	1 105	user	486 Mar 23 1985 LDEL20B.PUN
-rw-rw-r--	1 105	user	2592 Mar 27 1985 LDEL5.INP
-rw-rw-r--	1 105	user	486 Mar 28 1985 LDEL5.PUN
-rw-rw-r--	1 105	user	2592 Oct 23 1985 ORGCS4.INP
-rw-rw-r--	1 105	user	486 Oct 24 1985 ORGCS4.PUN
-rw-rw-r--	1 105	user	2592 Oct 23 1985 ORGCS6.INP
-rw-rw-r--	1 105	user	486 Oct 24 1985 ORGCS6.PUN
-rw-rw-r--	1 105	user	2592 Oct 23 1985 ORGCS8.INP
-rw-rw-r--	1 105	user	486 Oct 24 1985 ORGCS8.PUN
-rw-rw-r--	1 105	user	2592 Oct 17 1985 ORGCSC.INP
-rw-rw-r--	1 105	user	486 Oct 18 1985 ORGCSC.PUN

-rw-rw-r--	1 105	user	2592 Jun 14 1985 SK20G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 19 1985 SK20G50C.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK20G50D.INP
-rw-rw-r--	1 105	user	2592 Jun 4 1985 SK23G50.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK23G506.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK23G507.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK23G508.INP
-rw-rw-r--	1 105	user	2592 Jun 5 1985 SK23G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 4 1985 SK24G50.INP
-rw-rw-r--	1 105	user	2592 Jun 12 1985 SK24G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 13 1985 SK24G50C.INP
-rw-rw-r--	1 105	user	2592 Jun 4 1985 SK25G50.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK25G506.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK25G507.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK25G508.INP
-rw-rw-r--	1 105	user	2592 Jun 14 1985 SK25G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 4 1985 SK26G50.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SK26G507.INP
-rw-rw-r--	1 105	user	2592 Jun 12 1985 SK26G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 14 1985 SK27G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 3 1985 SK28G50.INP
-rw-rw-r--	1 105	user	2592 Jun 12 1985 SK28G50B.INP
-rw-rw-r--	1 105	user	2592 May 31 1985 SK30G50.INP
-rw-rw-r--	1 105	user	2592 Jun 14 1985 SK30G50B.INP
-rw-rw-r--	1 105	user	2592 Jun 24 1985 SKEAS1.INP
-rw-rw-r--	1 105	user	2592 Jul 10 1985 SKEAS10.INP
-rw-rw-r--	1 105	user	486 Jul 10 1985 SKEAS10.PUN
-rw-rw-r--	1 105	user	2592 Jul 10 1985 SKEAS11.INP
-rw-rw-r--	1 105	user	486 Jul 10 1985 SKEAS11.PUN
-rw-rw-r--	1 105	user	2592 Aug 13 1985 SKEAS13.INP
-rw-rw-r--	1 105	user	2592 Aug 13 1985 SKEAS15.INP
-rw-rw-r--	1 105	user	2592 Aug 13 1985 SKEAS17.INP
-rw-rw-r--	1 105	user	486 Oct 17 1985 SKEAS17.PUN
-rw-rw-r--	1 105	user	2592 Jun 25 1985 SKEAS2.INP
-rw-rw-r--	1 105	user	486 Jun 25 1985 SKEAS2.PUN
-rw-rw-r--	1 105	user	2592 Jun 26 1985 SKEAS2B.INP
-rw-rw-r--	1 105	user	486 Jun 26 1985 SKEAS2B.PUN
-rw-rw-r--	1 105	user	2592 Jun 25 1985 SKEAS3.INP
-rw-rw-r--	1 105	user	486 Jun 26 1985 SKEAS3.PUN
-rw-rw-r--	1 105	user	2592 Jun 26 1985 SKEAS4.INP
-rw-rw-r--	1 105	user	486 Jun 26 1985 SKEAS4.PUN
-rw-rw-r--	1 105	user	2592 Jun 26 1985 SKEAS4B.INP
-rw-rw-r--	1 105	user	486 Jun 26 1985 SKEAS4B.PUN
-rw-rw-r--	1 105	user	2592 Jun 27 1985 SKEAS5.INP
-rw-rw-r--	1 105	user	486 Jun 27 1985 SKEAS5.PUN
-rw-rw-r--	1 105	user	2592 Jun 28 1985 SKEAS6.INP
-rw-rw-r--	1 105	user	486 Jun 28 1985 SKEAS6.PUN
-rw-rw-r--	1 105	user	2592 Jul 1 1985 SKEAS7.INP
-rw-rw-r--	1 105	user	486 Jul 1 1985 SKEAS7.PUN
-rw-rw-r--	1 105	user	2592 Jul 2 1985 SKEAS8.INP
-rw-rw-r--	1 105	user	486 Jul 3 1985 SKEAS8.PUN
-rw-rw-r--	1 105	user	2592 Jul 2 1985 SKEAS9.INP
-rw-rw-r--	1 105	user	486 Jul 4 1985 SKEAS9.PUN
-rw-rw-r--	1 105	user	2592 Jul 11 1985 SKEASL.INP

-rw-rw-r--	1 105	user	486 Jul 11 1985 SKEASL.PUN
-rw-rw-r--	1 105	user	2592 Jul 15 1985 SKEASL2.INP
-rw-rw-r--	1 105	user	2592 Jul 16 1985 SKEASL3.INP
-rw-rw-r--	1 105	user	2592 Jul 16 1985 SKEASL4.INP
-rw-rw-r--	1 105	user	2592 Jun 5 1985 SKFRC.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SKFRC1.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SKFRC10.INP
-rw-rw-r--	1 105	user	2592 Jun 14 1985 SKFRC2.INP
-rw-rw-r--	1 105	user	2592 Jun 20 1985 SKFRC3.INP
-rw-rw-r--	1 105	user	2592 Jun 21 1985 SKFRC4.INP
-rw-rw-r--	1 105	user	3078 Feb 5 1985 SLIDACC.FTN
-rw-rw-r--	1 105	user	3078 Feb 5 1985 SLIDVEL.FTN
-rw-rw-r--	1 105	user	972 Dec 5 1984 SSMGZ250.INP
-rw-rw-r--	1 105	user	972 Dec 10 1984 SSMGZ350.INP
-rw-rw-r--	1 105	user	2592 Jan 2 1985 TC340B.INP
-rw-rw-r--	1 105	user	2592 Mar 19 1985 TC35.INP
-rw-rw-r--	1 105	user	2592 Mar 19 1985 TC40.INP
-rw-rw-r--	1 105	user	2592 Feb 8 1985 TC550B.INP
-rw-rw-r--	1 105	user	2592 Jan 2 1985 TC650B.INP
-rw-rw-r--	1 105	user	2592 Jan 2 1985 TC750B.INP
-rw-rw-r--	1 105	user	486 Nov 15 1984 TDHCSGY1.FTN
-rw-rw-r--	1 105	user	4131 Nov 15 1984 TDHCSGY1.INP
-rw-rw-r--	1 105	user	486 Nov 26 1984 TDHCSGY2.FTN
-rw-rw-r--	1 105	user	4131 Nov 26 1984 TDHCSGY2.INP
-rw-rw-r--	1 105	user	486 Nov 15 1984 TDHCSGYM.FTN
-rw-rw-r--	1 105	user	486 Nov 21 1984 TDHCSGZ1.FTN
-rw-rw-r--	1 105	user	3807 Nov 23 1984 TDHCSGZ1.INP
-rw-rw-r--	1 105	user	486 Nov 21 1984 TDHCSGZ2.FTN
-rw-rw-r--	1 105	user	3807 Nov 21 1984 TDHCSGZ2.INP
-rw-rw-r--	1 105	user	486 Apr 29 1985 TDHCSGZ4.FTN
-rw-rw-r--	1 105	user	762 May 13 1985 TDHCSGZ4.LST
-rw-rw-r--	1 105	user	486 Apr 18 1985 TDSSMGZ1.FTN
-rw-rw-r--	1 105	user	486 Oct 19 1984 TDSSMGZ1.INP
-rw-rw-r--	1 105	user	486 Oct 24 1984 TDSSMGZ2.FTN
-rw-rw-r--	1 105	user	972 Nov 20 1984 TDSSMGZ2.INP
-rw-rw-r--	1 105	user	972 Dec 10 1984 TDSSMGZ3.FTN
-rw-rw-r--	1 105	user	486 Mar 19 1985 TDSSMGZ4.FTN
-rw-rw-r--	1 105	user	972 Mar 5 1985 TDSSMGZ4.INP
-rw-rw-r--	1 105	user	2592 Jun 14 1985 TEST1.FTN
-rw-rw-r--	1 105	user	4131 Jun 14 1985 TEST1.INP
-rw-rw-r--	1 105	user	486 Jun 18 1985 TEST1.PUN
-rw-rw-r--	1 105	user	4131 Jun 28 1985 TEST12CG.INP
-rw-rw-r--	1 105	user	486 Jul 2 1985 TEST12CG.PUN
-rw-rw-r--	1 105	user	162 Jun 17 1985 TEST1R.INP
-rw-rw-r--	1 105	user	486 Jun 18 1985 TEST1R.PUN
-rw-rw-r--	1 105	user	2592 Jun 18 1985 TEST2.FTN
-rw-rw-r--	1 105	user	4131 Jun 14 1985 TEST2.INP
-rw-rw-r--	1 105	user	486 Jun 19 1985 TEST2.PUN
-rw-rw-r--	1 105	user	162 Jun 17 1985 TEST2R.INP
-rw-rw-r--	1 105	user	486 Jun 18 1985 TEST2R.PUN
-rw-rw-r--	1 105	user	2592 Jun 19 1985 TEST3.FTN
-rw-rw-r--	1 105	user	4131 Jun 21 1985 TEST3.INP
-rw-rw-r--	1 105	user	486 Jun 23 1985 TEST3.PUN
-rw-rw-r--	1 105	user	162 Jun 21 1985 TEST3R.INP

-rw-rw-r--	1	105	user	486 Jun 23 1985 TEST3R.PUN
-rw-rw-r--	1	105	user	2592 Jun 27 1985 TEST4.FTN
-rw-rw-r--	1	105	user	4131 Jun 27 1985 TEST4.INP
-rw-rw-r--	1	105	user	486 Jul 2 1985 TEST4.PUN
-rw-rw-r--	1	105	user	4131 Jun 28 1985 TEST42CG.INP
-rw-rw-r--	1	105	user	486 Jul 2 1985 TEST42CG.PUN
-rw-rw-r--	1	105	user	162 Jun 27 1985 TEST4R.INP
-rw-rw-r--	1	105	user	486 Jul 2 1985 TEST4R.PUN
-rw-rw-r--	1	105	user	2025 Mar 29 1985 WHAMA.FTN
-rw-rw-r--	1	105	user	2025 Apr 18 1985 WHAMASM.FTN
-rw-rw-r--	1	105	user	2025 May 7 1985 WHAMRST.FTN
-rw-rw-r--	1	105	user	20736 May 13 1985 XALMOST.FTN
-rw-rw-r--	1	105	user	20736 May 13 1985 XALMOSTC.FTN
-rw-rw-r--	1	105	user	20736 May 13 1985 XALMOSTS.FTN
-rw-rw-r--	1	105	user	486 May 13 1985 XHCSGY1.FTN
-rw-rw-r--	1	105	user	4131 May 13 1985 XHCSGY1.INP
-rw-rw-r--	1	105	user	486 May 13 1985 XHCSGY2.FTN
-rw-rw-r--	1	105	user	4131 May 13 1985 XHCSGY2.INP
-rw-rw-r--	1	105	user	1474 Aug 8 1985 YZ11DEG.DTA
-rw-rw-r--	1	105	user	1474 Jul 29 1985 YZ5DEG.DTA
-rw-rw-r--	1	105	user	2592 Jul 30 1985 YZ5DEG.INP
-rw-rw-r--	1	105	user	1474 Jul 29 1985 YZ6DEG.DTA
-rw-rw-r--	1	105	user	2592 Jul 29 1985 YZ6DEG.INP
-rw-rw-r--	1	105	user	1474 Jul 29 1985 YZ7DEG.DTA
-rw-rw-r--	1	105	user	2592 Jul 30 1985 YZ7DEG.INP
-rw-rw-r--	1	105	user	402 Aug 8 1985 YZAS11DE.DTA
-rw-rw-r--	1	105	user	402 Jul 30 1985 YZAS5DEG.DTA
-rw-rw-r--	1	105	user	402 Jul 30 1985 YZAS6DEG.DTA
-rw-rw-r--	1	105	user	402 Jul 30 1985 YZAS7DEG.DTA
-rw-rw-r--	1	105	user	402 Aug 8 1985 YZASL1.DTA
-rw-rw-r--	1	105	user	402 Aug 8 1985 YZASL2.DTA
-rw-rw-r--	1	105	user	486 Jul 30 1985 YZASORG.DTA
-rw-rw-r--	1	105	user	486 Aug 12 1985 YZASROT.FTN
-rw-rw-r--	1	105	user	402 Aug 8 1985 YZAST10.DTA
-rw-rw-r--	1	105	user	402 Aug 8 1985 YZAST11.DTA
-rw-rw-r--	1	105	user	402 Aug 8 1985 YZAST12.DTA
-rw-rw-r--	1	105	user	402 Aug 12 1985 YZAST7.DTA
-rw-rw-r--	1	105	user	402 Aug 9 1985 YZAST8.DTA
-rw-rw-r--	1	105	user	402 Aug 9 1985 YZAST9.DTA
-rw-rw-r--	1	105	user	1474 Aug 8 1985 YZL1.DTA
-rw-rw-r--	1	105	user	1474 Aug 8 1985 YZL2.DTA
-rw-rw-r--	1	105	user	1474 Sep 3 1985 YZL2C.DTA
-rw-rw-r--	1	105	user	1474 Sep 3 1985 YZL2CF.DTA
-rw-rw-r--	1	105	user	402 Aug 19 1985 YZL3B.DTA
-rw-rw-r--	1	105	user	1474 Sep 3 1985 YZL3C.DTA
-rw-rw-r--	1	105	user	1474 Sep 3 1985 YZL3CF.DTA
-rw-rw-r--	1	105	user	1474 Sep 3 1985 YZL3CFF.DTA
-rw-rw-r--	1	105	user	972 Aug 26 1985 YZL4.DTA
-rw-rw-r--	1	105	user	1474 Aug 26 1985 YZL4CF.DTA
-rw-rw-r--	1	105	user	972 Jul 30 1985 YZORG.DTA
-rw-rw-r--	1	105	user	486 Sep 3 1985 YZROTATE.FTN
-rw-rw-r--	1	105	user	1474 Aug 8 1985 YZT10.DTA
-rw-rw-r--	1	105	user	1474 Aug 27 1985 YZT10C.DTA
-rw-rw-r--	1	105	user	1474 Aug 27 1985 YZT10CF.DTA

-rw-rw-r--	1 105	user	1474 Aug 8 1985 YZT11.DTA
-rw-rw-r--	1 105	user	1474 Aug 27 1985 YZT11C.DTA
-rw-rw-r--	1 105	user	1474 Aug 27 1985 YZT11CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 8 1985 YZT12.DTA
-rw-rw-r--	1 105	user	1474 Sep 3 1985 YZT12C.DTA
-rw-rw-r--	1 105	user	1474 Sep 3 1985 YZT12CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT1C.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT1CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT2C.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT2CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT3C.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT3CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT5C.DTA
-rw-rw-r--	1 105	user	1474 Aug 29 1985 YZT5CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 28 1985 YZT6C.DTA
-rw-rw-r--	1 105	user	1474 Aug 28 1985 YZT6CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 12 1985 YZT7.DTA
-rw-rw-r--	1 105	user	1474 Aug 28 1985 YZT7C.DTA
-rw-rw-r--	1 105	user	1474 Aug 28 1985 YZT7CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 9 1985 YZT8.DTA
-rw-rw-r--	1 105	user	402 Aug 19 1985 YZT8B.DTA
-rw-rw-r--	1 105	user	1474 Aug 28 1985 YZT8C.DTA
-rw-rw-r--	1 105	user	1474 Aug 28 1985 YZT8CF.DTA
-rw-rw-r--	1 105	user	1474 Aug 8 1985 YZT9.DTA
-rw-rw-r--	1 105	user	3564 Sep 17 1985 Z.FTN
-rw-rw-r--	1 105	user	2592 Sep 10 1985 Z1.INP
-rw-rw-r--	1 105	user	486 Sep 11 1985 Z1.PUN
-rw-rw-r--	1 105	user	2592 Oct 23 1985 Z1CS5.INP
-rw-rw-r--	1 105	user	486 Oct 28 1985 Z1CS5.PUN
-rw-rw-r--	1 105	user	2592 Oct 23 1985 Z1CS7.INP
-rw-rw-r--	1 105	user	486 Oct 28 1985 Z1CS7.PUN
-rw-rw-r--	1 105	user	2592 Oct 23 1985 Z1CS9.INP
-rw-rw-r--	1 105	user	486 Oct 29 1985 Z1CS9.PUN
-rw-rw-r--	1 105	user	2592 Oct 17 1985 Z1CSC.INP
-rw-rw-r--	1 105	user	486 Oct 22 1985 Z1CSC.PUN
-rw-rw-r--	1 105	user	2592 Sep 11 1985 Z2.INP
-rw-rw-r--	1 105	user	486 Sep 12 1985 Z2.PUN
-rw-rw-r--	1 105	user	2592 Sep 17 1985 Z3.INP
-rw-rw-r--	1 105	user	1539 Sep 18 1985 Z3.PUN
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3CSA.INP
-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3CSA.PUN
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3CSB.INP
-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3CSB.PUN
-rw-rw-r--	1 105	user	2592 Oct 10 1985 Z3CSBN1.INP
-rw-rw-r--	1 105	user	972 Oct 11 1985 Z3CSBN1.PUN
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3CSC.INP
-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3CSC.PUN
-rw-rw-r--	1 105	user	2592 Oct 16 1985 Z3CSC2.INP
-rw-rw-r--	1 105	user	972 Oct 23 1985 Z3CSC2.PUN
-rw-rw-r--	1 105	user	2592 Oct 11 1985 Z3CSCN1.INP
-rw-rw-r--	1 105	user	972 Oct 15 1985 Z3CSCN1.PUN
-rw-rw-r--	1 105	user	2592 Oct 10 1985 Z3CSCN2.INP
-rw-rw-r--	1 105	user	2592 Oct 8 1985 Z3CSCN3.INP
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3CSD.INP

-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3CSD.PUN
-rw-rw-r--	1 105	user	2592 Oct 9 1985 Z3CSE.INP
-rw-rw-r--	1 105	user	972 Oct 11 1985 Z3CSE.PUN
-rw-rw-r--	1 105	user	486 Oct 11 1985 Z3CSET.PUN
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3RH1.INP
-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3RH1.PUN
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3RH3.INP
-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3RH3.PUN
-rw-rw-r--	1 105	user	2592 Oct 4 1985 Z3RH5.INP
-rw-rw-r--	1 105	user	1539 Oct 7 1985 Z3RH5.PUN
-rw-rw-r--	1 105	user	2592 Nov 1 1985 Z3SB1.INP
-rw-rw-r--	1 105	user	1539 Nov 4 1985 Z3SB1.PUN
-rw-rw-r--	1 105	user	2592 Nov 1 1985 Z3SB2.INP
-rw-rw-r--	1 105	user	1539 Nov 4 1985 Z3SB2.PUN
-rw-rw-r--	1 105	user	486 Oct 15 1985 Z3T.PUN
-rw-rw-r--	1 105	user	486 Oct 15 1985 Z3T2.PUN
-rw-rw-r--	1 105	user	2592 Nov 5 1985 Z4.INP
-rw-rw-r--	1 105	user	1539 Nov 6 1985 Z4.PUN
-rw-rw-r--	1 105	user	2592 Jan 6 1986 ZAT1.INP
-rw-rw-r--	1 105	user	1539 Jan 7 1986 ZAT1.PUN
-rw-rw-r--	1 105	user	3564 Oct 4 1985 ZCS.FTN
-rw-rw-r--	1 105	user	3564 Oct 16 1985 ZCS2.FTN
-rw-rw-r--	1 105	user	3564 Nov 1 1985 ZSB2.FTN
-rw-rw-r--	1 root	sys	0 Aug 16 14:19 dir002